Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Rozpoznávání hudebních žánrů

Plzeň 2016 Kateřina Štollová

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 9. května 2016 .....

Kateřina Štollová

Poděkování

Ráda bych touto cestou poděkovala vedoucímu diplomové práce panu Ing. Kamilu Ekšteinovi, Ph.D. za cenné rady a trpělivost během konzultací. Dále děkuji všem, kteří byli ochotni se ve svém volném čase zúčastnit testování. V neposlední řadě děkuji své rodině a přátelům za podporu během doby vzniku této práce i celé doby studia.

. . . . . .

Kateřina Štollová

Abstract

The goal of this thesis was to design and implement an algorithm for automatic classification of musical genres of the digital audio recordings stored in MP3 format. The recognized genre can be later used to automatically adjust an equalizer of an audio chain. The reason to do so is that the equalizer requires different settings for different music pieces. This setting mainly differs according to the used instruments and styles in the recording. Therefore, the classification focuses on the used instruments that manifest themself in the spectral characteristics of the analysed audio signal.

Abstrakt

Účelem této práce bylo navrhnout a implementovat algoritmus pro automatické klasifikování hudebních žánrů nahrávek ve formátu MP3. Takto určený žánr může být později použit k automatickému nastavení ekvalizéru, protože pro většinu hudebních žánrů je potřeba ekvalizér nastavit jiným způsobem. Toto nastavení se liší zejména podle použitých hudebních nástrojů a stylu hry. Proto se klasifikace zaměřuje hlavně na použité nástroje, které se projevují ve spektrální charakteristice zvukového signálu.

Obsah

1 Úvod

Úlohou této práce je prozkoumat možnosti automatické klasifikace hudebních žánrů a vytvořit program, který bude danou nahrávku zařazovat do jednoho z předem definovaných hudebních žánrů.

Účel automatického rozpoznávání hudebních žánrů je zřejmý. V posledních letech roste počet digitálně uchovávaných nahrávek. Udržovat pořádek v rozsáhlých databázích není vždy v lidských silách. Informací je takové množství, že není možné všechny manuálně analyzovat a třídit. Zpracování nových nahrávek a jejich zařazování může být prováděno automaticky na základě analýzy obsahu.

Důvodem k vypracování této práce byl požadavek na možné automatické nastavení ekvalizéru. Ekvalizér umožňuje posluchači upravit (zvýraznit či potlačit) určité frekvence v nahrávce za účelem dosažení kvalitnějšího celkového vjemu. Ekvalizéry mají typicky několik přednastavených schémat pro různé hudební žánry – pro poslech metalu budou důležitější jiné frekvence než pro poslech jazzu. Automaticky rozpoznaný hudební žánr může být vstupem pro aktivaci vhodného nastavení.

Úloha automatické klasifikace hudebních žánrů je řešena přibližně od roku 2000. Od té doby bylo pro její řešení vyzkoušeno mnoho přístupů, od spektrální či rytmické analýzy až po hledání opakujících se vzorů. V této práci je z důvodu použití klasifikátoru pro nastavení ekvalizéru kladen důraz zejména na klasifikaci žánrů na základě spektrální analýzy.

Klasifikace hudebních žánrů není triviální úloha, zejména z toho důvodu, že se jedná o velmi subjektivní problém a mnoho skladeb bude patřit do více skupin. Hudebních žánrů existuje nepřeberné množství a nové neustále přibývají, proto se tato práce zaměřuje jen na několik nejrozšířenějších.

Kvůli subjektivní povaze určování hudebních žánrů budou výsledky klasifikátoru porovnávány s výsledky referenční skupiny lidí, kteří budou zařazovat testovací hudební nahrávky do jedné z předem definovaných skupin (hudebních žánrů).

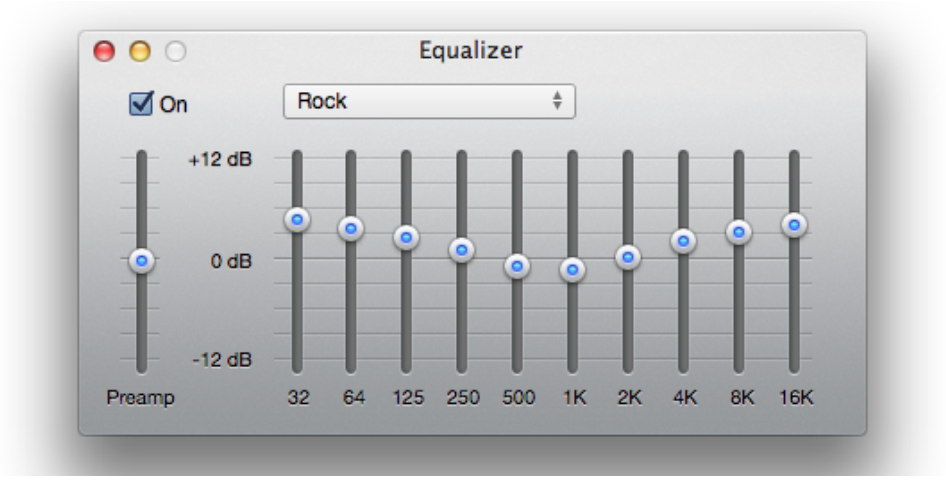
2 Teoretická část

V této kapitole budou postupně vysvětleny důležité teoretické pojmy pro pochopení úlohy automatického rozpoznávání hudebních žánrů. Nejprve vysvětlím pojmy související se samotným zadáním úlohy – roli ekvalizéru, jakým způsobem se liší některé hudební žánry či frekvenční rozsah běžně používaných nástrojů. Dále se zaměřím na zpracování digitálního signálu a možnosti extrakce příznaků pro účely rozpoznávání hudebních žánrů.

2.1 Teoretické pozadí úlohy

2.1.1 Ekvalizér

Ekvalizér je využíván při poslechu hudby zejména pro dosažení lepšího posluchačského zážitku. Jedná se o hardwarové či softwarové zařízení používané pro úpravu zvuku analogové či digitální hudební nahrávky. Příklad softwarového ekvalizéru zobrazuje obrázek 2.1.



Obrázek 2.1: Příklad softwarového klasifikátoru

Ekvalizér je sestava filtrů aplikovaná na signál v rozsahu vnímaným člověkem. Nejjednodušší ekvalizér má dva filtry, čímž umožňuje ovládat pouze vysoké frekvence a nebo basy. Složitější systémy pak mohou mít 3, 5 nebo 10 filtrů,

profesionální zařízení až 30. Větší počet filtrů dělí signál na více frekvenčních pásem a umožňuje ovlivňovat konkrétní frekvence přesněji [1].

Při ekvalizaci pak dochází k úpravě hlasitosti frekvencí v daném frekvenčním pásmu pomocí filtru. Frekvence kolem středu filtru jsou ovlivněny nejvíce, směrem k okrajům filtru vliv změny klesá.

Softwarové ekvalizéry nabízejí často několik přednastavených profilů dle hudebních žánrů – rock, pop, klasika, aj. Jednotlivé hudební žánry se od sebe často liší použitými hudebními nástroji, tedy zastoupenými frekvenčními pásmy, a k lepšímu zážitku z poslechu je třeba zvýraznit jiné frekvence.

Protože hudebních žánrů neustále přibývá, navíc se mezi s sebou překrývají, není možné je zastoupit v přednastavených profilech všechny. Přesto je vhodně vybraný profil (díky automaticky zvolenému žánru hudby) dobrým začátkem pro vlastní úpravy ekvalizéru.

2.1.2 Frekvenční rozsah hudebních nástrojů

Jak jsem zmínila dříve, jednotlivé hudební styly využívají různé nástroje, které se liší svým frekvenčním rozsahem. Rozsah konkrétních nástrojů se může mírně lišit v závislosti na ladění.

Frekvenční rozsah jednotlivých typů hudebních nástrojů přehledně zobrazuje tabulka 2.1 (inspirována [2] a [3]).

Tabulka zobrazuje přibližné fundamentální frekvence, které jsou jednotlivé nástroje schopné zahrát. Kromě těchto frekvencí se do výsledného díla navíc promítají rezonanční frekvence.

Příkladem je velký buben s fundamentální frekvencí mezi 60 – 100 Hz, kde má svou největší sílu. Díky harmonickým frekvencím se ale může projevit i ve vyšším frekvenčním pásmu, a to až 4 kHz.

2.1.3 Hudební žánry

Jak jsem již zmínila dříve, hudebních žánrů existuje velké množství. Pro svou práci jsem vybrala několik dostatečně rozšířených hudebních žánrů, které zde budu charakterizovat. Jednotlivé hudební žánry se liší použitými nástroji,

Kategorie (nástroje)	Nástroj	Frekvenční rozsah [Hz]
zpěv	ženský hlas	300 – 2500
	mužský hlas	100 – 900
dechové	flétna	250 – 2500
	klarinet	160 – 2500
	alt saxofon	160 – 850
	trubka	160 – 900
strunné	housle	200 – 3000
	violoncello	60 – 1000
	akustická kytara	80 – 900
	elektrická kytara	80 – 1000
	baskytara	40 – 300
bicí	činely	300 – 800
	malý buben	200 – 300
	velký buben	30 – 100
úderné	klavír	60 – 4000

Tabulka 2.1: Frekvenční rozsahy hudebních nástrojů

stylem, rytmem, ale i způsobem zpěvu. Vznikaly historicky s potřebou lidí pojmenovat nové směry v hudebním průmyslu [4].

Klasická hudba

Pojmem klasická hudba se dnes označuje hudba vznikající již od středověku. Největšího rozmachu dosáhla během baroka, ale existují i skladby mnohem starší. Klasická hudba obsahuje mnoho podkategorií: opera, orchestrální hudba, klavírní hudba, sborový zpěv, aj. Hlavními používanými nástroji jsou klavír, smyčcové nástroje nebo dechy.

Mezi nejznámější skladatele klasické hudby patří Johann Sebastian Bach či Wolfgang Amadeus Mozart. K novodobějším autorům lze pak zařadit Benjamina Brittena, který komponoval v druhé polovině 20. století a byl ovlivněn druhou světovou válkou.

Folk

Folk je hudební žánr, pro který jsou typické akustická kytara a výrazný zpěv. Často používá i další nástroje jako harmonika či různé perkusní nástroje. Autoři folkových písní si je většinou píší sami a vyjadřují jimi svůj postoj k světu. Mezi známé folkové zpěváky patří Jaromír Nohavica nebo Karel Kryl.

Jazz

Jazz má své kořeny v Americe v počátku 20. století a vychází z blues. Mezi hlavní používané nástroje patří dechové nástroje jako saxofon, trubka a pozoun. Dále je používán klavír a kontrabas. Jazz prošel lety vývoje a změn. V této práci jsem pod označením jazz uvažovala klasický jazz 20. století, ne novodobý jazz. Mezi jazzové interprety patří Louis Armstrong nebo Miles Davis.

Rap

Rap je hudební žánr spojený s afro-americkou kulturou a úzce souvisí s hip-hopem. Není používáno mnoho hudebních nástrojů, většina hudby je syntetizovaná. Písně jsou velmi rytmické a důležitou roli hraje rapování (rytmický přednes rýmujícího se textu) místo zpěvu. Mezi rapové zpěváky patří DMX nebo Eminem.

Rock

Rocková hudba patří mezi velmi populární hudební žánry. Její počátky se datují do 2. poloviny 20. let. Mezi používané hudební nástroje patří elektrická kytara, baskytara a bicí. Mezi autory rockové hudby patří např. kapely Queen či The Rolling Stones.

Metal

Metal je o něco mladší než rock, ze kterého vznikl. Stejně jako u rocku se používá zejména elektrická kytara, baskytara a bicí. Na rozdíl od rocku je metal agresivnější, co se zvuku i zpěvu týče. Příkladem metalových interpretů jsou Helloween a Disturbed.

2.2 Digitální signál

Automatické rozpoznávání hudebních žánrů patří mezi úlohy zpracování digitálního signálu. Obecně je signál fyzikální veličina, která přenáší určité množství informace. Nejčastěji je reprezentován jako proměnná závislá na čase. Signál spojitý v čase i amplitudě se nazývá analogový signál. Digitální signál je diskrétní v čase i amplitudě.

2.2.1 Digitalizace signálu

Pro potřeby zpracování signálu je třeba analogový signál převést na digitální. Proces převodu se nazývá digitalizace. Z analogového signálu jsou s periodou odpovídající vzorkovací frekvenci odečítány vzorky.

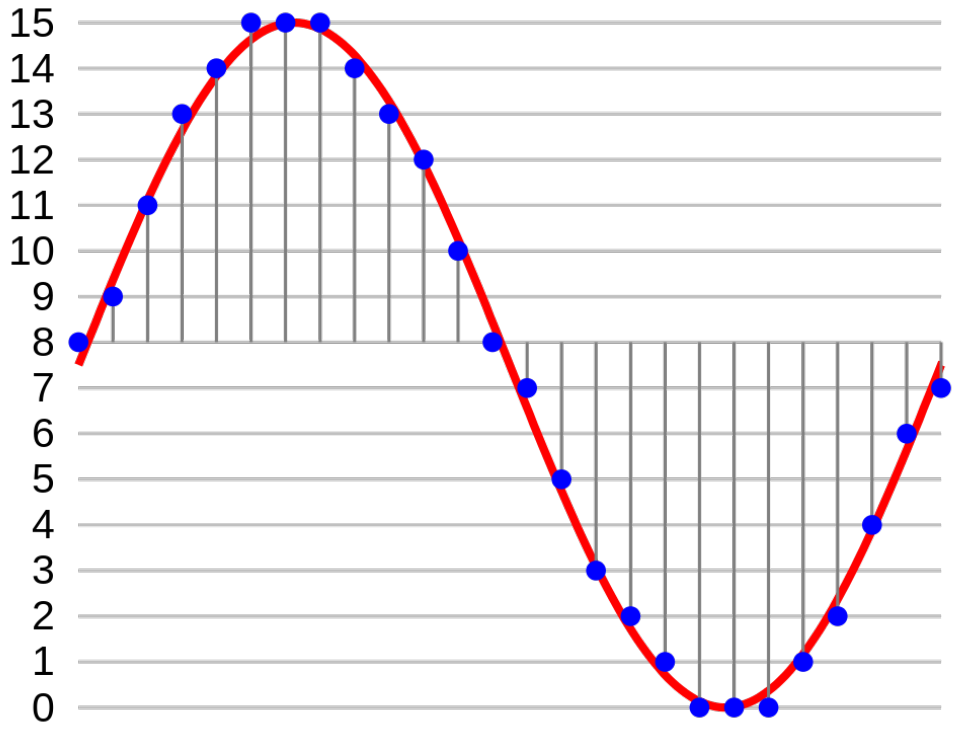
Pro zpracování pomocí počítače jsou jednotlivé vzorky uchovávány pomocí konečného počtu bitů a jejich hodnota tedy nebude vždy přesně odpovídat hodnotě v původním signálu.

Pro reprezentaci hodnot signálu se používá pulzně kódová modulace (*Pulse-code Modulation*), dále PCM. Na obrázku 2.2[1] je znázorněn příklad modulace sinusové vlny pro 4 bity. Jedná se pouze o ilustrační model, zvukové soubory jsou většinou modulovány vyšším počtem bitů.

Pásmo hodnot, kterých nabývá amplituda signálu, se rozdělí na několik menších pásem podle počtu bitů použitých pro modulaci – pro 4 bity to bude 16 pásem.

Hodnota odečtená během vzorkování je pak vyjádřena jako binární číslo dle

[1] Autorem obrázku je Aquegg, CC-BY-SA-3.0, Wikimedia Commons



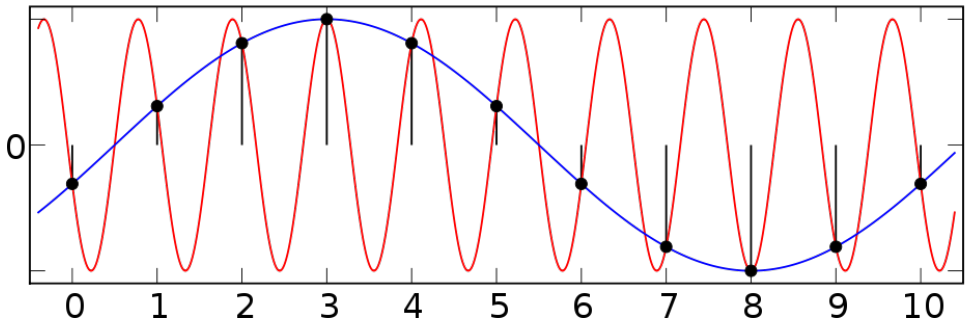
Obrázek 2.2: Příklad PCM pro 4 bity

pásma, do kterého spadá. Na obrázku je vidět, že některé hodnoty se v původním signálu liší, přesto jsou po modulaci vyjádřené stejně.

Minimální hodnotu vzorkovací frekvence stanovuje Nyquistův teorém, který říká, že rekonstrukce signálu je možná pouze v případě, když je vzorkovací frekvence větší než dvojnásobek maximální frekvence vyskytující se v signálu.

$$f_s > 2 \cdot f_{max}. \tag{2.1}$$

Frekvence $f_s/2$ se nazývá Nyquistova frekvence. Při porušení Nyquistova teorému hrozí při rekonstrukci signálu aliasing [5].



Obrázek 2.3: Znázornění špatně zvolené vzorkovací frekvence

Efekt aliasingu zobrazuje obrázek 2.3[2]. Je vidět, že původní červený signál je vzorkován špatnou frekvencí a při jeho rekonstrukci může být vytvořen zvuk o úplně jiné frekvenci.

2.2.2 Vlastnosti zvuku

Z fyzikálního hlediska je zvuk mechanické vlnění, jehož zdrojem je chvění pružných těles. Frekvence zvukového signálu slyšitelného pro člověka se pohybuje mezi 16 Hz a 20 kHz, přičemž nejvyšší citlivost je na frekvence 2 kHz – 4 kHz. Vlnění s vyšší frekvencí je ultrazvuk, s nižší frekvencí je infrazvuk.

Hudba je složena z jednoduchých a složených tónů. Tóny charakterizuje intenzita, výška a barva.

Výška tónu je určena jeho frekvencí, kdy vyšší frekvence odpovídá vyššímu tónu. Intenzitu člověk vnímá jako hlasitost zvuku. Zvuky stejné frekvence i intenzity se mohou lišit barvou. Ta je určena počtem vyšších harmonických tónů, jejich frekvencemi a amplitudami. Složené tóny se liší podle způsobu vzniku v různých oscilátorech, tj. v oscilátorech s různou velikostí, tvarem apod. Díky barvě tónu rozeznáváme různé hudební nástroje.

Další charakteristiky zvuku zmiňuje ve své práci Guaus [6]. Jedná se zejména o dlouhodobější charakteristiky. Mezi střednědobé charakteristiky zařazuje Guaus rytmus, melodii a harmonii.

[2]Autorem obrázku je Moxfyre, CC-BY-SA-3.0, Wikimedia Commons

Dimenze	Charakteristika	Popis
Krátkodobá	Frekvence	Výška tónu
	Intenzita	Hlasitost
	Barva	Barva tónu (dle způsobu vzniku)
Střednědobá	Rytmus	Opakující se vzory
	Melodie	Sekvence tónů
	Harmonie	Akordy
Dlouhodobá	Skladba	Celková organizace hudebního díla

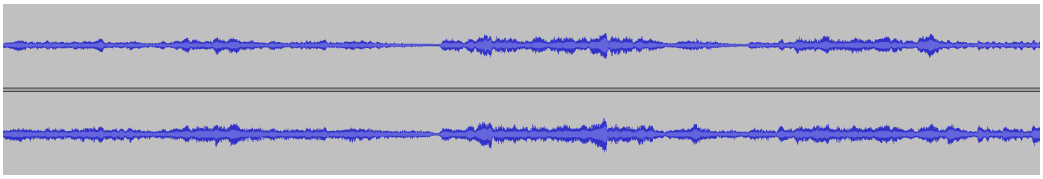
Tabulka 2.2: Rozdělení zvukových charakteristik

Rytmus souvisí s periodickým opakováním určitých vzorů v čase. Melodie je tvořena sekvencí jednotlivých tónů. Harmonie se skládá z více současně znějících tónů, tj. jedná se o vícezvuky, akordy.

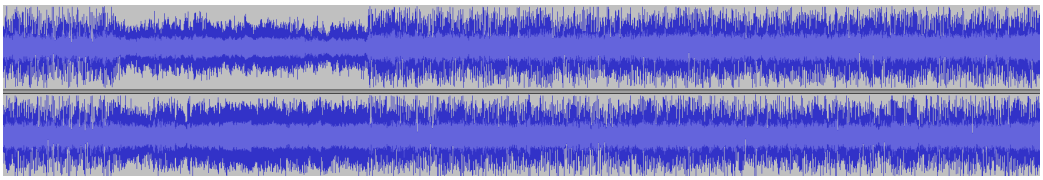
Jako dlouhodobou hudební charakteristiku zmiňuje Guaus skladbu, jako celkový pohled na hudební dílo, zejména opakování témat, pauzy, změny temp apod. Výše zmíněné charakteristiky shrnuje tabulka 2.2.

2.2.3 Oblasti zpracování digitálního signálu

V časové doméně je digitální signál zaznamenáván jako proměnná závislá na čase. Pro zpracování se často převádí do frekvenční domény, kde lze pozorovat jednotlivé frekvence, ze kterých se signál skládá.



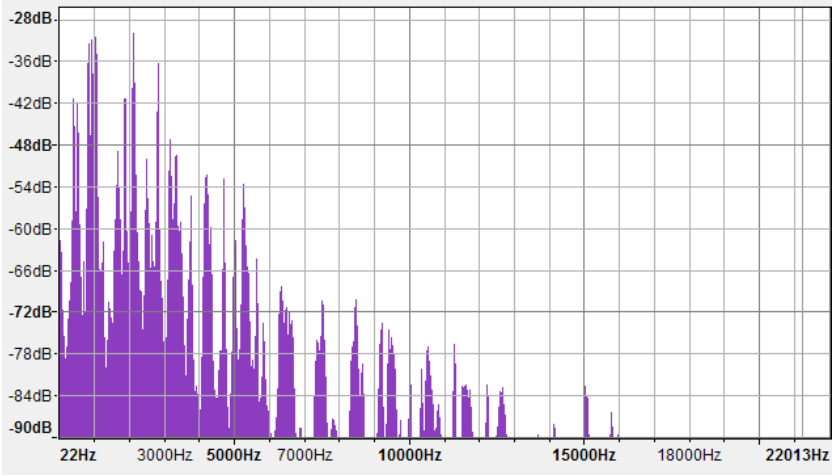
(a) Klasická skladba v časové doméně



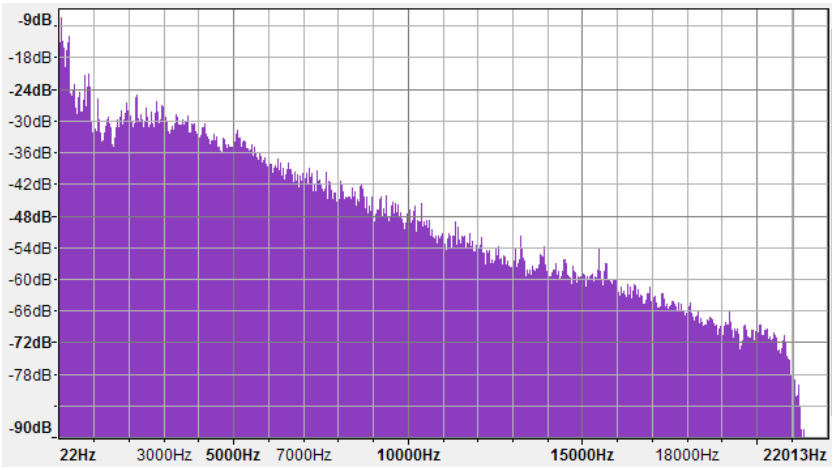
(b) Metalová skladba v časové doméně

Obrázek 2.4: Klasická a metalová skladba v časové doméně

Obrázky 2.4a a 2.4b zobrazují klasickou a heavy-metalovou skladbu v časové doméně. Pro srovnání, obrázky 2.5a a 2.5b zobrazují tytéž skladby ve frekvenční doméně, resp. odhad jejich výkonové spektrální hustoty.



(a) Odhad výkonové spektrální hustoty klasické skladby



(b) Odhad výkonové spektrální hustoty metalové skladby

Obrázek 2.5: Klasická a metalová skladba ve frekvenční doméně

V tomto případě lze pozorovat, které frekvence se ve skladbách nejvíce projevují a které se v nich nevyskytují vůbec.

Pro převod z časové domény do frekvenční se používá diskrétní Fourierova transformace (*Discrete Fourier Transformation*), dále DFT. Za určitých podmínek lze použít její rychlejší variantu, rychlou Fourierovu transformaci (*Fast Fourier Transformation*), dále FFT.

2.3 Metody zpracování digitálního signálu

Existuje několik metod, kterými lze zkoumaný signál připravit pro další analýzu. Zejména se jedná o metody pracující s krátkodobými charakteristikami signálu.

2.3.1 Diskrétní Fourierova transformace

Fourierova transformace je matematická metoda, která pomocí integrální transformace převádí signál do frekvenční domény (jedná se o vyjádření funkce v jiné bázi) a vyjadřuje jej pomocí periodických bázových funkcí sin a cos.

DFT je definována dle rovnice (2.2) pro $k = 0, 1, 2, ..., N - 1$

$$X_{DFT}[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi nk/N}. \tag{2.2}$$

Jiný náhled na DFT poskytuje Lyons [7]. Ten přepisuje rovnici DFT pomocí Eulerovo vztahu na

$$X_{DFT}[k] = \sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi nk}{N}\right) - i \sum_{n=0}^{N-1} x[n] \sin\left(\frac{2\pi nk}{N}\right) \tag{2.3}$$

a pohlíží na ni jako korelaci zkoumaného signálu s komplexní sinusoidou o frekvenci odpovídající k/N. Tato korelace se chová jako přizpůsobený filtr (*Matched Filter*).

Frekvence jednotlivých pásem závisí na vzorkovací frekvenci f_s a celkového počtu vzorků N a pro n-tý vzorek se vypočte dle (2.4)

$$f(n) = \frac{nf_s}{N}. \tag{2.4}$$

2.3.2 Rychlá Fourierova transformace

FFT je způsob, jak vypočítat DFT části signálu efektivněji než pouhou implementací vzorce (2.2). Zatímco v případě výpočtu DFT je časová složitost výpočtu $O(n^2)$, užitím FFT se časová složitost sníží na $O(n \cdot \log(n))$, což znamená pro větší n zásadní zrychlení výpočtu. Podmínkou pro dosažení této složitosti je, aby počet požadovaných koeficientů n byl mocninou 2.

Algoritmus Colley-Tukey

Existuje několik algoritmů pro výpočet FFT. Mezi nejznámější patří algoritmus Colley-Tukey (*Cooley–Tukey Algorithm*), který využívá symetrie DFT, kdy hodnota $X_{DFT}[k + iN] = X_{DFT}[k]$ pro všechna celá i viz (2.5)

$$\begin{aligned} X_{DFT}[N+k] &= \sum_{n=0}^{N-1} x[n] e^{-i2\pi n(N+k)/N} \\ &= \sum_{n=0}^{N-1} x[n] e^{-i2\pi n} e^{-i2\pi nk/N} \\ &= \sum_{n=0}^{N-1} x[n] e^{-i2\pi nk/N}. \end{aligned} \tag{2.5}$$

Rovnice využívá platnosti $e^{i2\pi n} = 1$ pro jakékoliv celé číslo n. Cooley a Tukey rozdělí N koeficientů, pro které se výpočet provádí, na dvě poloviny – sudé a liché – dle (2.6).

$$\begin{aligned} X_{DFT}[k] &= \sum_{n=0}^{N-1} x[n] e^{-i2\pi nk/N} \\ &= \sum_{n=0}^{N/2-1} x[2m] e^{-i2\pi(2m)k/N} + \sum_{n=0}^{N/2-1} x[2m+1] e^{-i2\pi(2m+1)k/N} \\ &= \sum_{n=0}^{N/2-1} x[2m] e^{-i2\pi mk/(N/2)} + \sum_{n=0}^{N/2-1} x[2m+1] e^{-i2\pi mk/(N/2)}. \end{aligned} \tag{2.6}$$

Rozdělená rovnice se skládá ze dvou výpočtů $N \cdot (N/2)$, což stále odpovídá časové složitosti $O(N^2)$. Nicméně právě zde je nyní využita výše zmíněná

symetrie. Pro každou část rovnice stačí počítat pouze polovinu výpočtů. Pro dosažení časové složitosti $O(n \log(N))$ je použit princip rozděl a panuj, kdy se dělení na dvě části opakuje rekurzivně tak dlouho, dokud je m sudé [5].

2.3.3 Spektrální hustota

Při převodu signálu z časové do frekvenční domény získáme odhad spektrální výkonové hustoty (*Power Spectral Density Estimate*), dále pro jednoduchost spektrální hustoty, což je kvantitativní vyjádření zastoupení energie jednotlivých frekvencí v signálu.

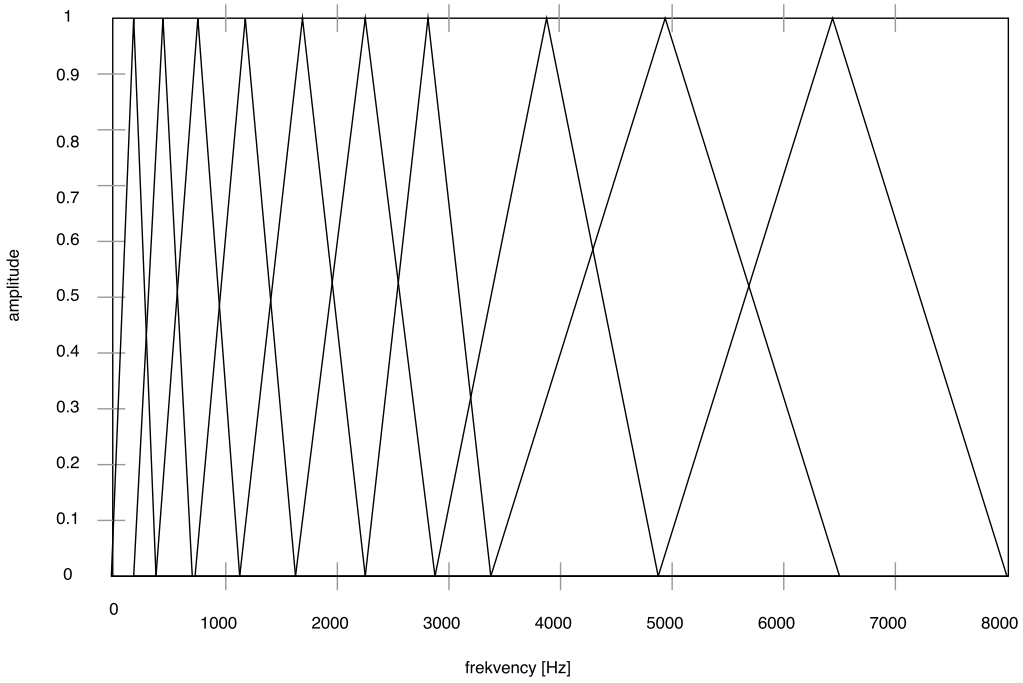
Se spektrální hustotou souvisí barva zvuku. Ta se liší dle původu signálu a projevuje se právě ve tvaru obálky spektrální hustoty. V rámci zpracovávání zvukového signálu pro účely rozpoznávání se pro reprezentaci tvaru obálky spektrální hustoty používají různé metody, mezi než patří i mel-frekvenční kepstrální koeficienty (*Mel Frequency Cepstral Coefficients*), dále MFCC. Jejich vysvětlení a způsob extrakce ze signálu je popsán v [8].

MFCC

Extrakce MFCC ze zvukového signálu probíhá v několika krocích. Nejprve se signál rozdělí do krátkých časových úseků, po kterých se bude zpracovávat. Pro rozpoznávání řeči je to typicky 20 – 40 ms. Tato časová okna by se měla alespoň částečně překrývat.

1. Pro každé časové okno se vypočte spektrum, např. pomocí FFT.
2. Na spektrum se aplikují trojúhelníkové filtry.
3. V každém filtru se integruje energie signálu a z výsledku se vypočte logaritmus.
4. Provede se diskrétní kosinová transformace (*Discrete Cosine Transformation*), dále DCT.

DCT se používá z důvodu, že jednotlivé trojúhelníkové filtry se navzájem překrývají a vypočtené koeficienty spolu korelují. DCT spočtené příznaky



Obrázek 2.6: Příklad 10 trojúhelníkových filtrů

dekoreluje. Při použití pro rozpoznávání řeči se používají typicky koeficienty 2 – 13.

První koeficient zahrnuje první komponentu FFT, která odpovídá průměru všech komponent a jejíž hodnota se obvykle pohybuje blízko nuly. Po převodu do melovy stupnice se jedná o velmi malé záporné číslo, které neodpovídá hodnotám ostatních koeficientů.

Z obrázku 2.6 je vidět, že rozmístění trojúhelníkových filtrů ani jejich velikost není konstantní. Melova stupnice je logaritmická a klade důraz na frekvence, které vnímá člověk nejvíce. Pro přepočet Herzů do melovy stupnice se používá vzorec (2.7), pro opačný přepočet pak slouží vzorec (2.8).

$$M(f) = 1125 \ln(1 + \frac{f}{700}), \tag{2.7}$$

$$M^{-1}(m) = 700(e^{\frac{m}{1125}} - 1). \tag{2.8}$$

Někdy se jako doplňkové příznaky používají první a druhá derivace MFCC tzv. ΔMFCC a ΔΔMFCC, které udávají spektrální změny pro jednotlivé koeficienty. První derivace se vypočte jako

$$\Delta_x[m,t] = c_x[m,t] - c_x[m,t-1], \tag{2.9}$$

kde $c_x[m,t]$ představuje m-tý koeficient v čase t. Druhá derivace pak dle rovnice (2.10).

$$\Delta\Delta_x[m,t] = \Delta_x[m,t] - \Delta_x[m,t-1]. \tag{2.10}$$

2.4 Strojové učení

Úloha automatického rozpoznávání hudebních žánrů patří mezi úlohy strojového učení. Strojové učení je proces, kdy systém modifikuje svou strukturu či chování tak, aby v budoucnu dosahoval lepších výsledků.

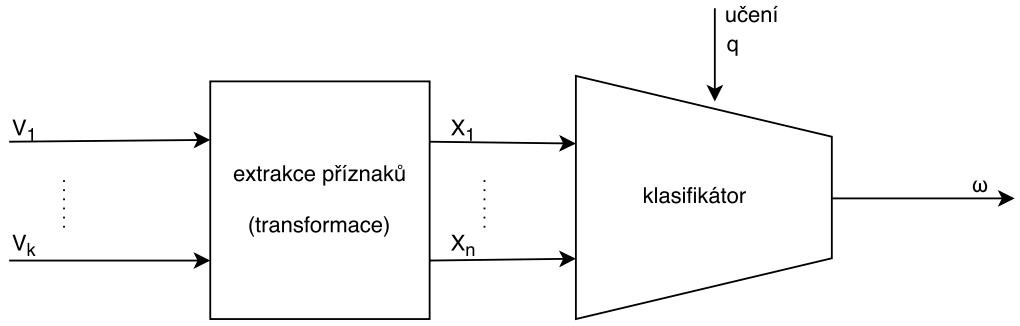
Strojové učení lze rozdělit na dvě velké skupiny, a to

- učení s učitelem, kdy má systém při učení informaci o správném řešení pro trénovací data a
- učení bez učitele, kdy systém nemá žádnou další informaci.

Strojového učení lze v dnešní době aplikovat na celou škálu problémů. Jednou možností aplikace je klasifikace, která je používána v této práci.

2.4.1 Klasifikační úloha

Klasifikace je proces, kdy systém na základě vstupu rozhodne o příslušnosti testovaného vzorku do jedné z definovaných tříd. Vstupem klasifikační úlohy jsou data určená ke klasifikaci, výstupem pak třída, ke které testovaný vzorek patří.



Obrázek 2.7: Obecná klasifikační úloha

Na obrázku 2.7 je zobrazena obecná klasifikační úloha. Z dat na vstupu $\mathbf{v}$ jsou extrahovány příznaky $\mathbf{x}$, které vstupují do klasifikátoru. Klasifikátor na základě svého nastavení q, které bylo zajištěno učením, určí příslušnost vzorku ke třídě. Matematicky lze toto zapsat

$$\omega = d(\mathbf{x}, q), \tag{2.11}$$

kde ω je identifikátor třídy, vektor $\mathbf{x}$ je obraz objektu, q je nastavení klasifikátoru a funkce d představuje rozhodovací pravidlo.

Úloha patří mezi algoritmy učení s učitelem. Proto je potřeba připravit dostatečně velkou a reprezentativní skupinu trénovacích dat, která budou mít příznak s informací, do které třídy náleží. Na této množině dat bude klasifikátor natrénován a bude zajištěno jeho ideální nastavení q.

2.4.2 Algoritmy strojového učení

Pro klasifikační úlohu lze použít několik algoritmů strojového učení. V následujících odstavcích jsou popsány dva z nich, které byly použity v diplomové práci. Jedná se o klasifikaci pomocí umělé neuronové sítě, dále NN, a algoritmus k-nejbližších sousedů (*K-nearest Neighbors*), dále k-NN.

Umělá neuronová síť

Umělá neuronová síť je adaptivní systém pro strojové určení, který během učení mění svoji strukturu. Skládá se z umělých neuronů, které jsou propojeny

pomocí synapsí a uspořádány do vrstev.

Výhodami neuronové sítě jsou vysoká generalizační schopnost a široká uplatitelnost na různé problémy. Mezi nevýhody patří velmi pomalý proces učení a množství dat, která jsou potřeba pro natrénování sítě.

Neuronová síť pracuje na principu šíření signálu a aktivace jednotlivých neuronů. Každý neuron je propojen s neurony v předchozí vrstvě, od kterých přijímá signály. Spojení (synapse) mezi neurony mají určitou váhu, která je během učení upravována tak, aby došlo k nejlepšímu nastavení sítě pro daný problém.

Každý vstup neuronu je vynásoben hodnotou váhy svého spojení. Všechny příchozí signály jsou sečteny a vstupují do aktivační funkce. Výstupem této funkce je aktivace neuronu, která je poslána do další vrstvy sítě.

Aktivační funkce mohou být různé pro různé úlohy. Měly by však být spojité a diferencovatelné v oboru reálných čísel. Velmi často se jako aktivační funkce používá sigmoida definovaná dle rovnice (2.12).

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{2.12}$$

Typicky se vícevrstvá neuronová síť skládá ze tří vrstev: vstupní vrstvy, skryté vrstvy a výstupní vrstvy. Množství neuronů ve vstupní vrstvě odpovídá počtu vstupů, ve výstupní vrstvě pak počtu výstupů. Pro klasifikaci do n tříd může mít neuronová síť až n výstupů, kdy aktivace n-tého neuronu bude odpovídat příslušnosti vzorku do n-té třídy.

Množství neuronů ve skryté vrstvě se může lišit dle typu úlohy a charakteru dat. Obecně platí, že síť s více neurony ve skryté vrstvě bude mít lepší výsledky, ale každá úloha má svou optimální hranici.

Algoritmy učení NN

Nejpoužívanější algoritmus učení umělé neuronové sítě je algoritmus zpětného šíření (*Backpropagation Algorithm*). Na základě trénovacího vzorku na vstupu se aktivují všechny neurony v síti. Dle aktivace výstupní vrstvy a očekávané odpovědi se určí chyba, na jejímž základě se pak zpětně po vrstvách upraví váhy jednotlivých neuronů.

Algoritmus zpětného šíření má nedostatky při úpravě vah jednotlivých neuronů – může vyžadovat příliš mnoho kroků, případně oscilovat kolem správného řešení. Toto odstraňuje algoritmus *RPROP* [9].

Na rozdíl od algoritmu zpětného šíření, který váhy jednotlivých synapsí upravuje v závislosti na velikosti chyby, *RPROP* upravuje hodnoty vah vždy o předem určenou hodnotu Δ_{ij} pro i-tý neuron v j-té vrstvě.

Během učení je velikost Δ_{ij} adaptivně upravována dle chyby dle jednoduchých pravidel:

- vždy když chyba vypočtená pro váhu w_{ij} změní znaménko (znamení, že přírůstek je příliš velký a algoritmus překročil minimum), hodnota Δ_{ij} je snížena,
- když znaménko chyby zůstává stejné, hodnota Δ_{ij} se o něco zvýší, aby došlo k urychlení konvergence.

Když jsou upraveny všechny hodnoty Δ_{ij}, dojde k jejich aplikování k příslušným vahám, a to

- přičtení, pokud derivace chyby je kladná nebo
- odečtení, pokud derivace chyby je záporná.

K-NN

K-NN je jednoduchý algoritmus používaný pro regresi a klasifikaci. Při klasifikaci se v příznakovém prostoru najde k nejbližších sousedů testovaného vzorku. Vzorek je poté přiřazen ke třídě, ke které patří většina těchto jeho sousedů.

Pro učení ideálního k se používá tzv. křížová validace (*K-fold Cross-validation*). Algoritmus křížové validace používaný v této práci je definován následovně:

1. Nastav $k = 1$,
2. rozděl náhodně trénovací data do n skupin,

3. vezmi 1 skupinu jako testovací $(1/n)$, zbytek tvoří referenční množinu $((n-1)/n)$,
4. poveď klasifikaci pomocí k-NN pro všechny testovací vzorky a vypočti chybu,
5. pokud $k < K_{\max}$ inkrementuj k a opakuj od kroku 2.

Hodnota k s nejmenší chybou se posléze zvolí jako parametr pro klasifikátor.

2.5 Extrakce příznaků

Pro klasifikaci je nejprve nutné z digitálního signálu extrahovat příznaky, které daný vzorek budou dostatečně vhodně reprezentovat. Příznaky tvoří příznakový vektor, který je projekcí vzorku v příznakovém prostoru.

Účelem extrakce příznaků je zvolit příznaky tak, aby vzorky, které se velmi liší, měly i velmi odlišné příznakové vektory (budou od sebe vzdálené v příznakovém prostoru). Naopak podobné vzorky by měly mít podobné příznakové vektory (jejich vektory budou v příznakovém prostoru blízko).

Tzanetakis a Cook ve své práci [10] ukazují na možné metody pro extrakci příznaků z akustického signálu pro účely rozpoznávání hudebních žánrů. Používají tři různé přístupy – spektrální příznaky, rytmické příznaky a příznaky synchronizované s periodou signálu.

Extrakci příznaků pro účely klasifikace hudebních žánrů se ve své práci věnuje i Guaus [6], který porovnává různé spektrální a rytmické charakteristiky digitálního signálu.

2.5.1 Spektrální příznaky

Jako spektrální příznaky používají Cook a Tzanetakis příznaky, které se běžně používají pro rozpoznávání řeči. Protože spektrální příznaky jsou ve své podstatě krátkodobé a pro rozpoznávání hudebních žánrů je třeba spíše dlouhodobější charakteristiky, používají Cook a Tzanetakis strukturní a analytické časové okno.

Strukturní okno je větší časový úsek, který by měl odpovídat době, která je třeba k rozpoznání hudebního žánru, resp. struktury hudebního díla. Strukturní okno se skládá z menších analytických oken, pro které se počítají níže uvedené spektrální vlastnosti. V rámci strukturního okna se pak používá jejich statistické vyjádření, např. pomocí střední hodnoty a odchylky. Cook a Tzanetakis používají strukturní okno o délce 1 s a analytické okno o délce 23 ms.

Spektrální centroid

Spektrální centroid je definován jako střední hodnota frekvence signálu vážená amplitudou spektra. Nejedná se o průměrnou frekvenci – různé nástroje budou mít různý spektrální centroid, i když budou hrát stejný tón. V čase t se vypočítá dle (2.13)

$$C_t = \frac{\sum_{n=1}^{N} M_t[n] \cdot f_n}{\sum_{n=1}^{N} M_t[n]}, \tag{2.13}$$

kde $M_t[n]$ je amplituda koeficientu FFT n v čase t a f_n je jeho frekvence.

Spektrální rolloff

Spektrální rolloff je definován jako hodnota R_t, pro kterou platí rovnice (2.14)

$$\sum_{n=1}^{R_t} M_t[n] = T_h \cdot \sum_{n=1}^{N} M_t[n], \tag{2.14}$$

kde hodnota T_h je typicky na rozmezí 0.8 – 0.95. Stejně jako spektrální centroid je spektrální rolloff používán pro vyjádření tvaru obálky výkonové spektrální hustoty.

Spektrální tok

Spektrální tok je definovaný jako kvadrát hodnoty rozdílu normalizované frekvence $N_t[n]$ n-tého koeficientu v čase t a v čase $t-1$ (2.15). Jedná se

tedy o měřítko lokálních spektrálních změn v čase.

$$F_t = \sum_{n=1}^{N} (N_t[n] - N_{t-1}[n])^2. \tag{2.15}$$

Mel-frekvenční kepstrální koeficienty

Jako další příznaky jsou velmi hojně používány MFCC, zejména kvůli dobré úspěšnosti při užití v rozpoznávání řeči. Tzanetakis a Cook používají pouze první 5 MFCC, zatímco Guaus používá 12 MFCC. V obou pracích se do příznakového vektoru zapisuje střední hodnota a odchylka pro daný koeficient.

2.5.2 Rytmické příznaky

Při určování hudebního žánru může být užitečná informace týkající se rytmu skladby. Skladba se skládá z taktů, které mohou být různého typu dle rytmu – nejznámějším je takt ¾. Horní číslo udává počet dob v taktu, dolní pak hodnotu kolika dobám odpovídá celá nota.

Pro rytmickou charakteristiku skladby zjišťujeme počet dob za minutu (*Beat per Minute*), dále BPM. Existuje několik algoritmů pro detekci dob, resp. zjištění BPM.

Beat histogram

Jedná se o koncept, který ve své práci představili Cook a Tzanetakis [10]. Vstupní signál je zpracováván v časové oblasti. Signál se nejprve pomocí vlnové transformace (*Wavelet Transformation*) rozdělí na několik části dle frekvencí. Pro každou část se provedou následující kroky:

1. Extrakce obálky signálu (*Full Wave Rectification*).
$$y[n] = |x[n]|. \tag{2.16}$$
2. Aplikování dolní propusti (*Low-pass Filter*) kvůli vyhlazení signálu. Hodnota parametru $\alpha = 0.99$.
$$z[n] = (1 - \alpha)y[n] + \alpha z[n - 1]. \tag{2.17}$$

3. Převzorkování pro dosažení lepšího výkonu v pozdějším výpočtu.

$$a[n] = z[kn], \tag{2.18}$$

   kde k je konstanta. Cook a Tzanetakis navrhují ve své práci $k = 16$.

4. Odečtení střední hodnoty.

$$b[n] = a[n] - E[a[n]]. \tag{2.19}$$

Po aplikování těchto kroků na všechny části signálu se signály integrují a provede se autokorelace (2.20).

$$c[n] = \frac{1}{N} \sum_{n} b[n]b[n-k]. \tag{2.20}$$

Další postupy

Pro detekci dob lze využít i výše zmíněný spektrální tok (2.15), který zachycuje změny ve spektrální hustotě.

Pro většinu skladeb platí, že rytmus vnímaný člověkem většinou udávají nástroje s nižší fundamentální frekvencí jako jsou bicí, bas kytara či kontrabas. Spektrální tok budeme proto počítat jen pro frekvenční pásmo, které odpovídá těmto nástrojům.

Výskyt doby se projeví vyšší hodnotou spektrálního toku (došlo ke změně ve sledovaných frekvencích), naopak nižší hodnoty spektrálního toku budou znamenat konstantní hodnoty frekvencí a nejsou zajímavé [6].

Jako příznaky z rytmického histogramu se používají perioda BPM, relativní amplituda první doby, poměr první a druhé doby [10]. Další autoři používají počet dob, jejich distribuci, maximální a minimální amplitudy apod. [6].

3 Realizace

Kapitola popisuje celkový průběh řešení. Věnuje se odůvodnění provedených rozhodnutí, samotné implementaci i problémům, ke kterým během vývoje došlo. Dále je v této kapitole podrobně popsán program pro automatické rozpoznávání hudebních žánrů, který jsem v rámci diplomové práce vytvořila.

3.1 Hudební žánry

Protože měl být výstupem klasifikační program, bylo nejprve nutné určit počet skupin hudebních žánrů a jednotlivé žánry.

Zařazování skladeb dle stylů je značně subjektivní a různí lidé se často nemohou shodnout o charakteristikách jednotlivých žánrů. Důvodem je pravděpodobně to, že žánrů existuje velké množství a některé se mohou navzájem překrývat.

Pro tuto úlohu jsem se rozhodla použít jen ty nejznámější hudební žánry, u kterých nebude problém určit jejich charakteristiku. Počet skupin jsem se snažila udržet co nejmenší. Zejména kvůli tomu, aby se jednotlivé skupiny mohly co nejvíce lišit. Dalším důvodem byla menší trénovací množina a zkrácení doby trénování. Počet skupin není problém v budoucnu navýšit bez větších úprav kódu.

V prvním návrhu jsem zvolila sedm hudebních žánrů, a to *klasickou hudbu*, *country a folk*, *elektronickou hudbu*, *rap a hip-hop*, *jazz a swing*, *rock* a *metal*. Volila jsem známé žánry, pod kterými by si každý měl představit hudební styl či interpreta.

Pop jsem úmyslně vynechala, protože se mi zdál příliš obecný a jeho hlavní rysy těžko identifikovatelné. Jako pop se totiž dá označit jakákoliv momentálně populární hudba.

Po několika testech a porovnání nastavení ekvalizéru pro elektronickou hudbu a pro rap, jsem se rozhodla tyto dvě skupiny sloučit, byly si navzájem dost podobné.

Dále jsem se rozhodla odstranit country. Zejména kvůli sporům ohledně označení country u nás a v Americe. Americké country má blíže k rocku, zatímco české je spíše podobné folku.

Ve finální práci je tedy šest skupin hudebních žánrů, a to následující.

1. Klasická hudba,
2. folk,
3. elektronická hudba a rap,
4. jazz a swing,
5. punk, rock a
6. metal.

Osobně považuji všechny tyto žánry jako dobře rozlišitelné. Nicméně jsem během testu referenční skupiny zjistila, že hranice mezi nimi není pro každého jasná a pro různé lidi může ležet jinde. To se ukázalo zejména mezi metalem a rockem či rockem a folkem.

3.2 Trénovací data

Trénovací i testovací data jsou vybrané skladby z online zdrojů, které nabízejí hudbu pod licencí *Creative Commons*, konkrétně *Free Music Archive* a *SoundCloud*.

Ideální by bylo použít již vytvořený trénovací set s rozřazenými skladbami podle hudebních žánrů, bohužel jsou tyto hůře dohledatelné či licencované.

FMA – Free music archive [11] je webová databáze nabízející množství hudebních skladeb ke stažení. Většina skladeb je pod různými licencemi Creative Commons. Na stránce lze vyhledávat podle žánrů, interpreta či podle požadované licence. Většina trénovacích dat pochází z této databáze.

SoundCloud [12] je webová komunitní stránka určená pro hudebníky a skladatele, kteří zde najdou prostor pro zveřejnění svého díla. Většina skladeb na stránce však není volně ke stažení a je třeba dávat pozor na licenci, pod kterou autor své dílo zveřejnil. Tento zdroj byl použit jen jako doplňkový. Zejména z toho důvodu, že nelze jednoduše dohledat skladby podle licence.

3.2.1 Popis datasetů

Popis trénovacího datasetu přehledně zobrazuje tabulka 3.1. Průměrná délka nahrávek na žánr je 1 hodina 30 minut. Celková délka všech skladeb je 9 hodin. Všechny nahrávky jsou ve formátu MP3, 2 kanálové a vzorkované frekvencí 44100 Hz.

Žánr	Počet skladeb	Celková délka
1 – klasická hudba	24	1 hod 29 min 36 s
2 – folk	29	1 hod 28 min 41 s
3 – elektro, rap	24	1 hod 30 min 59 s
4 – jazz, swing	21	1 hod 31 min 45 s
5 – rock, punk	25	1 hod 28 min 27 s
6 – heavy-metal	23	1 hod 30 min 23 s

Tabulka 3.1: Popis testovacího datasetu

Ohledně zařazení jednotlivých nahrávek do skupin jsem se inspirovala žánrem, kterým byla skladba označena v online databázi, ze které jsem ji získala. Všechny jsem pak poslouchala a pokud mi nepřišly pro daný žánr dostatečně charakteristické, vyřadila jsem je.

Důvod je zřejmý, v trénovacím datasetu by měly být pouze typické vzorky od každého žánru, aby byly mezi jednotlivými skupinami dobře patrné rozdíly.

Pro testování jsem použila 48 nahrávek ve formátu MP3. Také tyto nahrávky byly 2 kanálové vzorkované frekvencí 44100 Hz. Jejich celková délka byla 2 hodiny 52 minut.

Hudební žánry u testovacího setu jsem nejprve odhadla sama pro účely alfa testů klasifikátoru. V pozdější fázi vývoje jsem nechala hudební žánry určit skupinu dobrovolníků a jako *správný* hudební žánr jsem zvolila ten, který získal největší počet hlasů. Tento test je podrobněji rozebrán v kapitole 4.

Počet vzorků v testovacím datasetu jsem musela zvolit jako kompromis mezi dostatečně velkou množinou pro ověření kvality řešení a takovou celkovou dobou, kterou jsou lidé ochotni strávit poslouchánim v rámci referenčního testu.

3.3 Volba příznaků

Na základě předchozí studie jsem vybrala příznaky, které budu používat pro klasifikaci. Rozhodla jsem se použít různé příznakové vektory a jejich výsledky v závěru porovnat.

V několika následujících odstavcích vysvětlím, jakým způsobem jsem se rozhodla nahrávky zpracovávat a jak vytvářím jednotlivé příznakové vektory.

3.3.1 Délka zpracovávaného signálu

Ze všeho nejdřív bylo třeba určit ideální délku signálu, ze kterého se bude rozpoznávat hudební žánr. Dle [13] jsou lidé schopni určit hudební žánr poměrně přesně na základě 250 ms vzorku. Rozhodla jsem se provést test, abych toto tvrzení ověřila.

V anechoické komoře jsem pomocí kvalitních sluchátek poslouchala neznámé hudební vzorky, u kterých bylo mým úkolem určit žánr. Zmiňovaná doba 250 ms mi přišla na přesné určení hudebního žánru příliš malá. U rozšířenějších žánrů nebo typických skladeb nebyl problém, ten nastával když byl testovaný vzorek spíše obecný.

Nakonec jsem stanovila ideální časové okno na 1 – 2 sekundy. Pro obě délky jsem testovala klasifikátor, porovnala výsledky a na jejich základě určila ideální délku okna zpracovávaného signálu. Lepší výsledky jsem dosáhla pomocí delšího zmiňovaného časového okna, tedy 2 sekund.

3.3.2 Způsob zpracování skladeb

Pro klasifikaci jsem se inspirovala strukturním a analytickým oknem, které používají Tzanetakis a Cook ([10], [13]). Každou skladbu jsem rozdělila na

2 sekundové testovací vzorky s vynecháním prvních 10 sekund kvůli případnému tichu či nežádoucímu šumu.

Tyto vzorky pak dělím na menší analytická okna, která se navzájem z poloviny překrývají. Pro úlohu rozpoznávání řeči, která je podobná rozpoznávání hudebních žánrů, se používá kratší časové okno, typicky 20 – 40 ms. Pro rozpoznávání hudebních žánrů je však třeba zachytit dlouhodobější charakter signálu.

Během testování jsem vyzkoušela různé velikosti analytických oken a nakonec jsem určila jako nejlepší délku 250 ms. Pro 2 sekundový vzorek to znamená 15 analytických oken, počítáme-li s polovičním překryvem. Pro každé analytické okno extrahuji příznaky a příznakový vektor pak tvořím pro celý vzorek statisticky z těchto hodnot.

Testovací nahrávka, u které je úkolem určit hudební žánr, se typicky skládá z několika 2 sekundových vzorků. Pro každý vzorek získám příznakový vektor a na jeho základě hudební žánr, do kterého je vzorek přiřazen. Hudební žánr pro celou nahrávku je pak zvolen majoritně na základě příslušnosti všech vzorků, ze kterých se skládá.

Díky tomuto přístupu nezáleží na tom, jak jsou jednotlivé testované nahrávky dlouhé. Dokonce pokud bude skladba obsahovat několik částí typově patřících k různým hudebním žánrům, mělo by pravidlo většinového výběru zajistit, že bude zvolen ten nejvhodnější.

3.3.3 Zvolené příznaky

Rozhodla jsem se použít několik příznakových vektorů a jejich výsledky porovnat. Pro zvolení používaného příznakového vektoru lze použít parametr při spuštění programu. Při vynechání parametru jsou zvoleny výchozí spektrální příznaky.

MFCC

První příznakový vektor je tvořen pouze MFCC. Pro rozpoznávání řeči se používají koeficienty 2–13, ale pro úlohu klasifikace hudebních žánrů je třeba podchytit i vyšší frekvence. Z toho důvodu používám koeficienty 2–21, které odpovídají frekvencím 66 – 1736 Hz.

MFCC počítám pro každé analytické okno a pro celý vzorek zprůměruji hodnoty každého koeficientu. Takto bude zachyceno rozložení hodnot jednotlivých koeficientů a zároveň se potlačí případný šum.

Příznakový vektor je tvořen 20-ti hodnotami.

Původně jsem předpokládala, že bude třeba použít více koeficientů, abych zahrnula všechny frekvence, které se v hudbě vyskytují. Nicméně se ukázalo, že vyšší frekvence nejsou pro klasifikaci hudebních žánrů zajímavé. Více o tomto poznatku obsahuje sekce 3.7.3.

Spektrální příznaky

Pro spektrální příznaky je příznakový vektor tvořen stejně jako v předchozím případě průměrnými hodnotami 20 MFCC pro analytická okna.

Navíc pro každé analytické okno počítám spektrální centroid a spektrální rolloff. Pro ně do příznakového vektoru zapisuji jejich střední hodnotu a odchylku. Příznakový vektor pro spektrální charakteristiku obsahuje 24 hodnot.

Rytmické příznaky

Jako rytmické příznaky jsem zvolila použití spektrálního toku. Zkoušela jsem i vytvoření rytmického histogramu dle Cooka a Tzanetakise, ale výsledky obou způsobů byly srovnatelné a použití spektrálního toku je jednodušší.

Protože pro 250 ms okna je pro každý vzorek 14 hodnot spektrálního toku, zapisuji je do příznakového vektoru všechny.

Všechny příznaky

Poslední možností použitého příznakového vektoru je kombinace všech předchozích. Obsahuje 20 průměrných hodnot pro MFCC 2–21, střední hodnoty a rozptyly pro spektrální centroid a spektrální rolloff a rytmický příznakový vektor.

Celková velikost tohoto příznakového vektoru je 38 hodnot.

3.4 Zvolený algoritmus učení

Jako učící se algoritmus jsem si zvolila neuronovou síť, zejména kvůli tomu, že má velmi dobrou generalizační schopnost a obecně dobrou úspěšnost. Dále existuje řada implementací neuronových sítí, které nabízejí široké možnosti parametrizace, nastavení trénovacích algoritmů a paralelní průběh trénování.

Neuronová síť ale vyžaduje značné množství dat a trénuje se mnohem déle než ostatní algoritmy strojového učení. Z toho důvodu jsem se rozhodla pro druhý klasifikační algoritmus, kterým bych při testování odhadovala správné nastavení parametrů a který by mi poskytoval rychlejší zpětnou vazbu.

Zvolila jsem jednoduchý a rychlý algoritmus *k-NN*, který jsem implementovala sama. Pro určení nejlepší hodnoty parametru k používám algoritmus křížové validace s testem na 1/10 trénovacích dat. Algoritmus *k-NN* byl skvělou volbou díky jednoduchosti a rychlosti běhu, nicméně dával o něco horší výsledky než neuronová síť.

Program jsem navrhla tak, aby oba učící algoritmy měly společného generického předka, v budoucnu nebude proto problém případně klasifikační algoritmus vyměnit za jiný.

3.5 Použité technologie

Pro implementaci jsem se rozhodla použít jazyk `C++`, a to z důvodu jeho přenositelnosti, rychlosti výpočtů a mojí zkušenosti s ním. Pro `C++` také existuje mnoho knihoven použitelných pro zpracování zvuku a implementujících algoritmy strojového učení.

Pro práci se soubory MP3 používám knihovnu **BASS**, pro práci se souborovým systémem knihovnu **dirent**. Dále používám knihovnu **FANN**, která poskytuje implementaci umělé neuronové sítě, a knihovnu **libMFCC** pro výpočet mel-frekvenčních kepstrálních koeficientů.

Pro testy poslechu hudebních žánrů jsem používala aplikaci Audacity.

3.5.1 Knihovna BASS

V programu využívám knihovnu BASS verze 2.4.11. Jedná se o multiplatformní knihovnu pro práci se zvukovými soubory. Knihovna podporuje různé zvukové formáty, včetně formátu MP3, který je používán v této práci. Pro nekomerční užití je knihovna poskytována zdarma. [14]

Práci s knihovnou zobrazuje kód 3.1.

```
1  float fft[1024];
2
3  BASS_Init(-1, 44100, BASS_DEVICE_NOSPEAKER, 0, NULL);
4  HSTREAM m_hsStream = BASS_StreamCreateFile(FALSE, pFileName, 0, 0,
       BASS_STREAM_DECODE | BASS_SAMPLE_FLOAT);
5
6  BASS_ChannelSetPosition(m_hsStream, BASS_ChannelSeconds2Bytes(m_hsStream,
       dTime), BASS_POS_BYTE);
7  while (BASS_ChannelIsActive(m_hsStream))
8  {
9      BASS_ChannelGetData(m_hsStream, fft, BASS_DATA_FFT2048);
10     float dPosition = (float) BASS_ChannelBytes2Seconds(m_hsStream,
       BASS_ChannelGetPosition(m_hsStream, BASS_POS_BYTE));
11     if (dPosition >= dMaxTime)
12       break;
13 }
14
15 if (m_hsStream)
16     BASS_StreamFree(m_hsStream);
17 BASS_Free();
```

Kód 3.1: Příklad práce s knihovnou BASS

Před začátkem práce se souborem je nutné knihovnu inicializovat voláním `BASS_Init`. V příkladu je výchozí nastavení pro konzolovou aplikaci – bude použito výchozí zvukové zařízení, vzorkovací frekvence 44100 Hz. Práce se zvukovým souborem začíná voláním `BASS_StreamCreateFile`. Zde je načten soubor `pFileName`.

V příkladu je dále ze souboru od pozice `dTime` do pozice `dMaxTime` spočtena FFT pro $N = 2048$, tedy 1024 vzorků (pokud není specifikováno jinak, imaginární část není vrácena), a načtena do proměnné `fft`. `BASS_StreamFree` uvolňuje načtený soubor, `BASS_Free` uvolňuje ostatní alokované zdroje.

3.5.2 Knihovna FANN

FANN je knihovna implementující vícevrstvou umělou neuronovou síť poskytovaná pod licencí *GNU LESSER GENERAL PUBLIC LICENSE*. Je multiplatformní s podporou více jak 20 programovacích jazyků. Knihovna nabízí několik algoritmů učení (RPROP, Quickprop) a široké možnosti parametrizace neuronové sítě. [15]

Jednoduchý příklad použití knihovny FANN zobrazuje kód 3.2. Nejprve se vytvoří struktura sloužící pro reprezentaci sítě. V příkladu je vytvořena síť se třemi vrstvami, vstupní vrstva má počet neuronů roven hodnotě `num_input`, skrytá vrstva má počet neuronů daný proměnnou `num_hidden` a počet neuronů udává proměnná `num_output`.

Pro skrytou a výstupní vrstvu se nastaví aktivační funkce na symetrickou sigmoidu. Pro trénování sítě se používá funkce `fann_train_on_file`, kde jednotlivé parametry jsou síť určena k trénování, jméno souboru s trénovací množinou, maximální počet iterací, počet iterací mezi výstupy a požadovaná chyba. Natrénovanou síť lze pro pozdější potřeby uložit do souboru voláním `fann_save`.

```
1  struct fann *ann;
2
3  ann = fann_create_standard(3, num_input, num_hidden, num_output);
4
5  fann_set_training_algorithm(ann, FANN_TRAIN_RPROP);
6  fann_set_activation_function_hidden(ann, FANN_SIGMOID_SYMMETRIC);
7  fann_set_activation_function_output(ann, FANN_SIGMOID_SYMMETRIC);
8
9  fann_train_on_file(ann, file, UINT_MAX, uiEpochsToReport, fGoalError);
10
11 fann_save(ann, fileNET);
12 fann_destroy(ann);
```

Kód 3.2: Příklad trénování neuronové sítě s knihovnou FANN

Pro následnou klasifikaci pak stačí načíst natrénovanou neuronovou síť ze souboru voláním `fann_create_from_file` a spustit `fann_run`. Parametry jsou naučená neuronová síť a vektor s hodnotami vstupů, návratovou hodnotou je excitace výstupní vrstvy. Příklad klasifikace zobrazuje kód 3.3.

```
struct fann *ann;
float *output;

ann = fann_create_from_file(fileNET);
output = fann_run(ann, input);

fann_destroy(ann);
```

Kód 3.3: Klasifikace s knihovnou FANN

3.5.3 Knihovna libMFCC

Pro extrakci MFCC používám knihovnu libMFCC [16], která je psaná v C. Práce s ní je velmi jednoduchá – knihovna poskytuje jednu funkci pro získání k-tého koeficientu ze zadané FFT signálu.

```
for (int iCoeff = 1; iCoeff <= iNumMFCC; iCoeff++)
{
  float val = (float) GetCoefficient(fft, SAMPLERATE, 48, 1024, iCoeff);
}
```

Kód 3.4: Použití knihovny libMFCC

Kód 3.4 zobrazuje smyčku pro získání koeficientů MFCC. Prvním parametrem je FFT daného signálu, následuje vzorkovací frekvence, počet použitých filtrů pro výpočet koeficientů, počet koeficientů FFT a pořadové číslo požadovaného koeficientu.

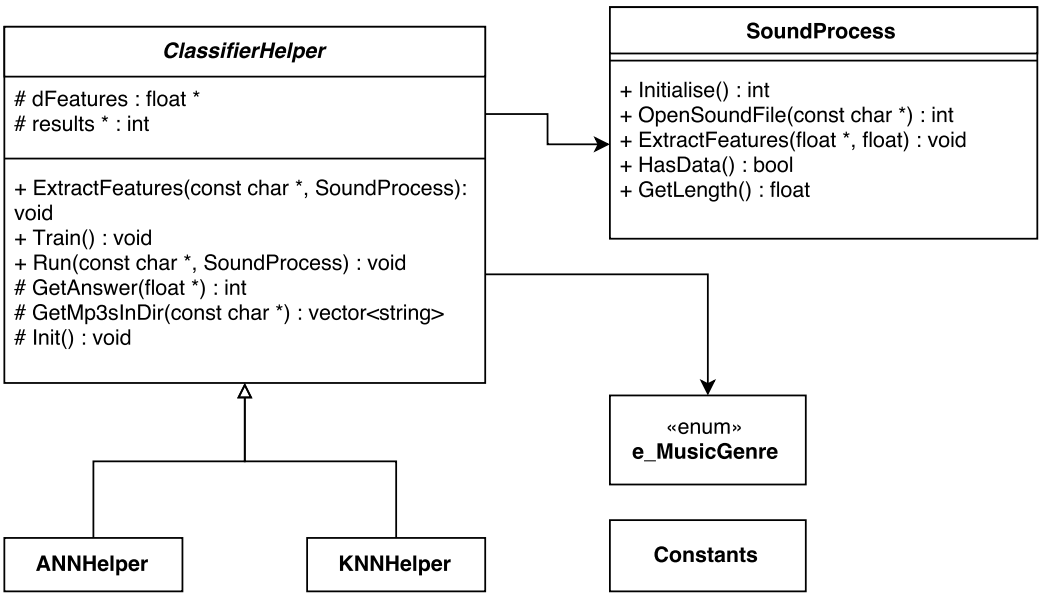
Knihovna byla vyvinuta speciálně pro rozpoznávání hudby, z toho důvodu jsem se ji rozhodla použít. Ve své práci popisují autoři, že knihovnu implementovali s přihlédnutím na rozpoznávání hudebních nástrojů [17].

Knihovna používá maximálně 48 navzájem se překrývajících filtrů, které pokrývají frekvence 0 – 9614 Hz. Střední frekvence pro k-tý filtr se vypočte dle vzorce (3.1).

$$f_c(k) = \begin{cases} 66.\overline{66}k & \text{pro } 1 < k < 14 \\ 1073.4(1.0711703)^{k-14} & \text{pro } 15 \leq k \leq 48 \end{cases} \tag{3.1}$$

3.6 Návrh programu

Strukturu programu jsem navrhla tak, aby bylo možné měnit parametry algoritmu i použitý algoritmus pomocí parametrů programu zadaných při spuštění. Obrázek 3.1 znázorňuje diagram tříd vytvořeného programu.



Obrázek 3.1: UML diagram tříd

Vytvořila jsem generickou třídu pro učení a klasifikaci `ClassifierHelper`, která obsahuje virtuální metody pro spuštění extrakce příznaků, trénování a klasifikaci. Její potomci (konkrétní klasifikátory) poté tyto metody budou implementovat. Lze tedy vytvořit více klasifikátorů a pomocí parametru zvolit, který bude použit.

Pro práci se zvukovým souborem slouží třída `SoundProcess`, která komunikuje s knihovnou `BASS` a udržuje handle na právě zpracovávaný zvukový soubor. Dále tato třída poskytuje jednotnou extrakci příznaků z otevřeného souboru, která je používána jak při extrakci příznaků, tak při klasifikaci.

Veškeré nastavení algoritmu je pak uloženo ve statické třídě `Constants`, jejíž hodnoty se plní při spuštění programu ze zadaných argumentů.

3.6.1 Logické rozdělení programu

Program se bude skládat ze tří logických celků:

- extrakce příznaků z dat a vytvoření trénovacího datasetu,
- trénování klasifikátoru a
- klasifikace.

Zejména kvůli časové náročnosti prvních dvou kroků nutných pro přípravu klasifikátoru jsem se rozhodla navrhnout program tak, aby jej bylo možné pustit pro každý logický celek zvlášť. Výstupy běhu extrakce příznaků a trénování jsou ukládány do souboru a používají je následující úseky programu.

V praxi to znamená, že lze pustit např. pouze extrakci příznaků, která vytvoří soubor s trénovacím datasetem pro pozdější použití.

Spuštění trénování a nebo klasifikace samozřejmě vyžaduje přítomnost výstupu z předcházejícího kroku. Pro trénování se jedná o zmíněný soubor s trénovacím datasetem, pro klasifikaci je to soubor s uloženou strukturou NN.

U klasifikace pomocí algoritmu k-NN se proces trénování spouští před spuštěním klasifikace, protože je vzhledem ke trénování neuronové sítě rychlý a jeho výstupem je nastavení pouze jednoho parametru. Z toho důvodu mi přišlo zbytečné vytvářet soubor s nastavením.

Struktura souboru s trénovacím datasetem

Struktura souboru s trénovacími daty je odvozená od formátu souboru, který vyžaduje knihovna FANN. Soubor začíná třemi celými čísly představujícími počet trénovacích vzorků v souboru, počet vstupů sítě (odpovídá velikosti příznakového vektoru) a počet výstupů (odpovídá počtu tříd). Dále následují jednotlivé trénovací vzorky – vždy řádek se vstupy následován řádkem s očekávaným výstupem sítě, viz 3.5.

```
3 3 1
0.5 1 -0.35
1
1 2 0
1
-1 -0.2 3
-1
```

Kód 3.5: Příklad souboru s trénovacími daty

Příklad formátu souboru obsahuje tři trénovací vzorky, příznakový vektor se skládá ze tří čísel a odpověď neuronové sítě bude jedno číslo.

3.6.2 Způsob zpracování dat

Kvůli urychlení trénování i klasifikace jsem se rozhodla pro dávkové zpracování jednotlivých trénovacích i klasifikovaných nahrávek.

K tomuto účelu využívám knihovnu **dirent**. Jako parametr při spuštění programu je očekávána cesta ke složce obsahující trénovací data v módu extrakce příznaků a/nebo cestu ke složce obsahující data ke klasifikaci v módu klasifikace.

Extrakce příznaků

Pro extrakci příznaků je program spuštěn s parametrem `F` a s cestou ke kořenové složce s trénovacími daty. Ta musí mít předepsanou strukturu, aby došlo ke správnému načtení všech dat. V kořenové složce jsou uloženy složky s čísly `01` – `06` pro používaných 6 hudebních žánrů. Pořadí hudebních žánrů odpovídá pořadí hodnot ve výčtovém typu `e_MusicGenre`.

Jednotlivé složky pak mohou obsahovat trénovací data ve formátu MP3. Počet souborů ve složkách nemusí být stejný a není nikde definován.

Klasifikace

Pokud je program spuštěn v módu klasifikace, tj. s parametrem `T`, je třeba, aby tento parametr následovala cesta ke složce obsahující testované nahrávky.

Všechny MP3 soubory v této složce pak klasifikátor postupně zpracuje a určí pro ně žánr.

Pro testování je dávkový běh užitečný, nicméně u klasického spuštění klasifikátoru se spíše očekává, že bude vstupem jedna testovaná nahrávka.

3.6.3 Třídy pro klasifikaci

ClassifierHelper

Jedná se o virtuální třídu, která slouží jako generický klasifikátor. Obsahuje virtuální metody `ExtractFeatures` pro extrakci příznaků a vytvoření trénovací množiny, `Train` pro trénování klasifikátoru a `Run` pro spuštění samotné klasifikace.

Metody `Train` a `Run` jsou specifické podle typu použitého klasifikátoru, proto jejich implementace obsahují až potomci třídy `ClassifierHelper`. Metoda `ExtractFeatures` používá pro získání příznaků třídu `SoundProcess`.

Dále třída obsahuje několik `protected` metod a atributů. Tyto metody jsou pomocné pro výpočty a používají je konrétní klasifikátory. Metoda `Init` slouží pro pozdní inicializaci klasifikátoru. Je v ní provedeno vše, co je nutné před spuštěním klasifikace, ale nemohlo být ještě provedeno v konstruktoru.

Metoda `GetMp3sInDir` slouží pro vytvoření seznamu cest ke všem souborům MP3 v zadané složce. Používá se jak pro extrakci příznaků, tak pro testování (metody `ExtractFeatures` a `Run`).

Metoda `GetAnswers` je abstraktní. Slouží na určení odpovědi na základě výstupu klasifikátoru, musí být tedy implementována pro každý klasifikátor specificky.

Atributy třídy jsou `dFeatures`, vektor příznaků pro právě zpracovávaný vzorek, a `results`, vektor sloužící pro test. Tento vektor v sobě uchovává informaci ohledně klasifikace jednotlivých 2 sekundových částí právě nahrávky, aby mohlo být následně rozhodnuto o jejím zařazení.

ANNHelper

Třída dědí od virtuální třídy `ClassifierHelper`. Jedná se o klasifikátor používající neuronovou síť, konkrétně implementaci **Artificial Neural Network Library** [15]. Tato implementace nabízí množství parametrů k nastavení modelu neuronové sítě, zejména několik algoritmů učení, různé aktivační funkce či různé metody výpočtu chyby.

Pro účely diplomové práce jsem použila neuronovou síť s jednou skrytou vrstvou. Počet neuronů ve vstupní vrstvě odpovídá velikosti příznakového vektoru (liší se na základě zvolených příznaků), výstupní má šest neuronů, což odpovídá počtu tříd. Dostatečný počet neuronů ve skryté vrstvě jsem po několika pokusech stanovila na 500.

Dále používám symetrickou sigmoidu jako aktivační funkci a pro trénování algoritmus *RPROP*. Příslušnost vzorku k n-té třídě se projeví aktivací n-tého neuronu ve výstupní vrstvě na hodnotu 1. Ostatní neurony budou mít hodnotu -1.

Knihovna nabízí i další algoritmy učení, např. *Quickprop* či klasické zpětné šíření. Většinu z nich jsem testovala a použitím *RPROP* se síť učila nejrychleji.

Přestože knihovna FANN umožňuje pustit trénování rovnou nad datovým souborem, rozhodla jsem se proces o něco více řídit. Zejména z toho důvodu, že pro rytmický příznakový vektor chyba sítě divergovala a trénování nebylo možné ukončit. Proces trénování sítě použitý v této práci zobrazuje kód 3.6.

Je vidět, že v případě překročení maximálního počtu iterací se kontroluje chyba *Fail Bit*, která představuje celkový počet bitů v dané iteraci pro všechny trénovací vzorky, které se lišily od předepsaného výstupu. Tyto konstanty jsem stanovila po pozorování několika úspěšných a neúspěšných běhů na 150000 iterací a 1000 bitů.

```
struct fann_train_data *data = fann_read_train_from_file(data_fileIN);

for (; ; epochs++)
{
  fann_train_epoch(ann, data);
  fThisError = fann_get_MSE(ann);
  uFailBits = fann_get_bit_fail(ann);

  if (fThisError < fGoalError || uFailBits == 0)  // standard stop, good
    break;
  else if (epochs % uiEpochsToReport == 0)
  {
    if (epochs > maxEpochs && uFailBits > epochThresh) // nonconverging
      break;
    // report
    printf("%d. Last error %lf, current error %lf. Fail bits %d\n", epochs,
    fLastError, fThisError, uFailBits);
    fLastError = fThisError;
  }
}
```

Kód 3.6: Proces trénování FANN

Pro všechna úspěšná trénování byla požadovaná chyba dosažena buď dříve než po 150000 iteracích, anebo touto dobou byl počet chybných bitů menší než 500. Tímto nastavením jsem tedy ovlivnila pouze takové trénovací procesy, jejichž chyba měla tendenci divergovat a zůstávala velká i po množství úprav struktury sítě.

KNNHelper

Třída implementuje druhý použitý algoritmus pro klasifikaci k-NN. Opět dědí od `ClassifierHelper`. Pro určení ideální hodnoty k, která představuje počet uvažovaných sousedů, se používá křížová validace.

Třída pro reprezentaci trénovacích i testovaných vzorků používá pomocnou strukturu `Sample`, která se skládá z příznakového vektoru, žánru, ke kterému vzorek náleží a vzdálenosti. Vzdálenost se plní při klasifikaci a vztahuje se vždy k právě klasifikovanému vzorku. Strukturu zobrazuje kód 3.7.

```
struct Sample
{
  std::vector<float> data;
  e_MusicGenre genre;
  float distace;
};
```

Kód 3.7: Struktura pro uchování dat

Pro načtení trénovací množiny slouží metoda `LoadSamples`. Během trénování se nastaví nejlepší hodnota k. Při klasifikaci se pak pro klasifikovaný vzorek počítá vzdálenost ke všem vzorkům v trénovací množině. Ty jsou pak seřazeny podle vypočtené vzdálenosti a na základě k nejbližších zvolena třída, do které bude testovaný vzorek zařazen. Eukleidovská vzdálenost mezi příznakovými vektory $\mathbf{x}$ a $\mathbf{y}$ se vypočítá dle vzorce

$$d = \sqrt{\sum_i (x[i] - y[i])^2}. \tag{3.2}$$

3.6.4 Ostatní třídy

Třída `SoundProcess`

Pro práci se zvukovým souborem je používána třída `SoundProcess`, která slouží jako adaptér knihovny **BASS** [14].

Práce se zvoleným souborem obsahujícím zvuk začíná jeho otevřením voláním metody `OpenSoundFile`, která otevře požadovaný soubor a uchová ukazatel na něj pro další práci.

Nejdůležitější metodou pro klasifikaci je metoda `ExtractFeatures`. Ta z otevřeného souboru dle nastavených parametrů programu extrahuje příznakový vektor. Parametry metody jsou adresa k uložení extrahovaných příznaků a čas, kde v souboru začíná aktuální vzorek.

Zpracování souboru probíhá typicky volání metody `ExtractFeatures` ve smyčce, viz kód 3.8. Soubor se programově rozdělí na vzorky o velikosti `textWSize`, které jsou zpracovány.

```
1 float dTime = 0.1f;
2 soundProcess->OpenSoundFile(fileName);
3 while (dTime + textWSize <= soundProcess->GetLength())
4 {
5   soundProcess->ExtractFeatures(dFeatures, dTime);
6   dTime += textWSize;
7 }
```

Kód 3.8: Zpracování zvukového souboru

Třída obsahuje dále několik pomocných metod, např. `GetTimbralFeatures` či `GetMFCC`, které slouží, jak názvy napovídají, pro získání MFCC a spektrálních příznaků, a jsou využívány v metodě `ExtractFeatures`.

Třída `Constants`

Protože jednotlivé parametry pro klasifikaci jsou konstantní pro jeden běh a využívají se na více místech, rozhodla jsem se vytvořit statickou třídu `Constants`, která bude pro daný běh programu uchovávat hodnoty všech parametrů.

Nejdůležitějšími parametry jsou `m_dSampleWindow`, který představuje velikost zpracovávaného vzorku, a `m_dAnaWindow` představující velikost analytických oken. Tyto parametry slouží k rozdělení signálu pro zpracování. Dalším důležitým parametrem je použitý příznakový vektor `m_eFeatures`.

Třída obsahuje tři metody – `GetFeaturesCount` pro získání velikosti příznakového vektoru dle zvolených příznaků, `GetFeaturesType`, která vrací textový popis zvoleného příznakového vektoru, a pomocnou `GetWindowsCount`, která vrací počet analytických oken ve vzorku.

Výčtový typ `eUsedFeatures`

Slouží pro reprezentaci použitého příznakového vektoru. Obsahuje následující hodnoty:

- `Features_MFCC_only` pro použití pouze MFCC,
- `Features_Timbral_Texture` pro spektrální příznakový vektor,
- `Features_Rhytmic` reprezentující rytmické příznaky,
- `Features_Both` pro kompletní příznakový vektor.

Výčtový typ `e_MusicGenre`

Představuje jednotlivé hudební žánry, resp. třídy pro klasifikaci. Pro rozšíření počtu tříd, není nutné upravovat zbytek programu, stačí přidat hodnotu před `MusicGenre_Count_of_Elements`.

V aktuální verzi programu obsahuje výčet následující hodnoty:

- `MusicGenre_Classic`,
- `MusicGenre_Folk`,
- `MusicGenre_Rap`,
- `MusicGenre_Jazz`,
- `MusicGenre_Rock`,
- `MusicGenre_Metal` a
- `MusicGenre_Count_of_Elements` pro snadné zjištění počtu tříd.

3.7 Problémy

V následujících odstavcích popíši problémy, na které jsem narazila během vývoje. Také se zmíním o slepých uličkách při realizaci práce.

3.7.1 Problémy s konvergencí neuronové sítě

V některých případech, kdy trénovací data byla špatně rozmístěna v příznakovém prostoru, docházelo k velmi pomalé konvergenci neuronové sítě či dokonce její divergenci.

Standardní funkce pro trénování neuronové sítě `fann_train_on_file` se zastavuje pouze v případě dosažení požadované chyby či po maximálním počtu iterací. Protože jsem počet iterací neomezovala, v případě špatné konvergence se trénování nezastavilo.

Standardní postup pro ukončení běhu nekonvergujícího trénování je založen na kontrole rozdílu chyb aktuální a předchozí iterace. Tento postup bohužel kvůli divergenci nefungoval, takže jsem musela zvolit postup náhradní.

Na základě pozorování úspěšných a neúspěšných (nekonvergujících) běhů trénování neuronové sítě, jsem se rozhodla stanovit hranici počtu iterací, po jejímž dosažení kontroluji chybu výstupu jednotlivých neuronů, tzv. *Fail Bit*. Pokud hodnota této chyby není pod stanovenou hranicí, prohlásím algoritmus za nekonvergující a jeho běh ukončím.

3.7.2 Rytmické příznaky

Pro extrakci rytmických příznaků ze signálu jsem vyzkoušela obě popsané metody, nicméně žádný nepřinesl pozitivní výsledky.

Možným důvodem je, že jsem pro implementaci rytmického histogramu nepoužila na rozdělení signálu do různých frekvenčních pásem vlnovou transformaci, ale využila jsem FFT.

Protože jsem FFT aplikovala na 250 ms okna, je možné že jsem příliš zmenšila počet hodnot použitých pro výpočet rytmického histogramu a ten pak neodpovídal signálu, ale jeho hrubé aproximaci.

Druhý způsob zkoumající změny ve spektrálním toku pro nižší frekvence měl o něco lepší výsledky, proto jsem se ho rozhodla nakonec použít. Nicméně i tak je příznakový vektor zaměřující se na rytmus nejméně úspěšným pro klasifikaci.

3.7.3 Počet MFCC

Jak jsem již zmiňovala, pro klasifikaci hudebních žánrů kvůli zastoupení vyšších frekvencí jsem očekávala nutnost použití více mel-frekvenčních koeficientů než se běžně používá pro rozpoznávání řeči. Některé hudební nástroje dosahují frekvencí až 16 kHz.

Dle [17] je možné ale dosáhnout dobrých výsledků i použitím 20-ti MFCC, které pokrývají pouze frekvenční pásmo 66 – 1736 Hz, což je přibližně desetina předpokládaného rozsahu.

Rozhodla jsem se provést test pro ověření důležitosti jednotlivých frekvenčních pásem v různých hudebních žánrech.

V programu Audacity jsem na vybrané skladby aplikovala nejprve dolní propust, jejíž pomocí jsem kompletně odřízla frekvence nad 2000 Hz, poté jsem na ně aplikovala horní propust, kterým jsem naopak odřízla frekvence pod 2000 Hz. Při poslechu byl originálu více podobný vzorek s nižšími frekvencemi. Pro některé hudební žánry, jako rock a metal, se dokonce zdůraznily jejich typické rysy.

Přes příznivé výsledky poslechového testu jsem se rozhodla vyzkoušet použití jak 20 MFCC tak 48 MFCC, které pokrývají o něco frekvenční pásmo 66 – 9614 Hz. Při srovnání jejich výsledků jsem si nevšimla žádného zhoršení – oba testy dopadly srovnatelně. Doba trénování a testování byla samozřejmě značně kratší pro 20 koeficientů.

Na základě těchto testů jsem usoudila, že i když testované vzorky obsahují vyšší frekvence, ty nenesou tolik informace důležité pro klasifikaci jako frekvence nižší. Z toho důvodu používám pouze 20 MFCC.

4 Výsledky

Kapitola se věnuje průběhu jednotlivých testů a shrnuje výsledky práce. Výstupy klasifikátoru jsou porovnávány s daty získanými testem referenční skupiny lidí.

4.1 Průběh testování

Program prošel několika vlnami testů. V první části vývoje jsem program testovala proti množině dat, kterou jsem sama rozřadila do skupin. Účelem tohoto testování bylo zjištění ideálního nastavení parametrů jednotlivých algoritmů. Druhá vlna testování probíhala proti datům, která rozřadila referenční skupina lidí. Výsledky těchto testů jsou rozebrány na následujících stranách.

4.1.1 Test referenční skupiny

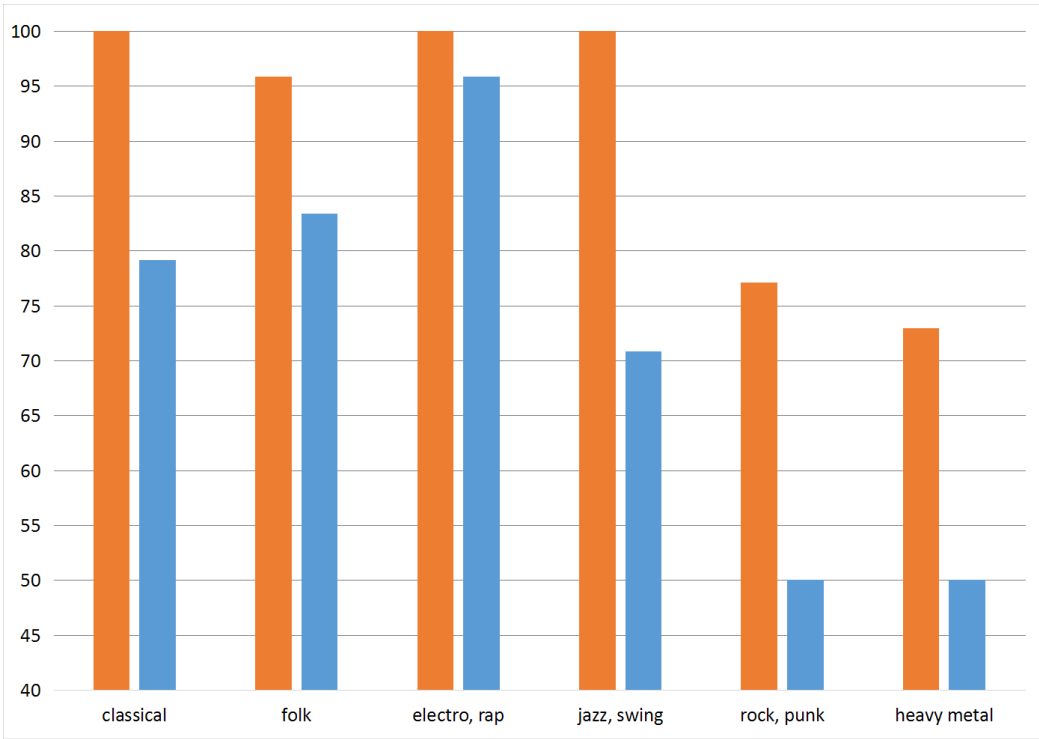
Protože je úloha rozpoznávání hudebních žánrů velmi subjektivní a na výsledcích se mnohdy neshodnou ani dva lidé, rozhodla jsem se provést test na větší skupině lidí. Na základě výsledku tohoto testu jsem pak rozhodla o správném zařazení jednotlivých testovaných nahrávek.

Celkem se testu účastnilo 24 lidí ve věku od 12 do 59 let. Medián věku účastníků byl 25 let. Žen v testovací skupině bylo devět a mužů patnáct. Většina účastníků o sobě prohlásila, že má kladný vztah k hudbě, i když ne všichni měli zkušenost se všemi testovanými hudebními žánry.

Každý účastník testu dostal k dispozici 48 nahrávek, u nichž bylo jeho úkolem určit žánr, a to právě jeden z mnou definovaných. Kompletní výsledky testu zobrazují tabulky A.1 a A.2, které jsem vzhledem k jejich rozměrům umístila do přílohy. Zde uvedu jen nejzajímavější poznatky z testu.

Statistické shrnutí výsledků testu zobrazuje obrázek 4.1. Pro zařazení testované skladby do žánru jsem brala většinový hlas. Ne vždy se však na daném žánru shodli všichni účastníci testu, většina tedy nemusí odpovídat 100%.

Graf zobrazuje statistiku výsledků testu. Jedná se o procentuální hodnoty počtu účastníků testu, kteří hlasovali pro žánr, který zvítězil (tedy danou většinu). Oranžový sloupec představuje medián procentuálního vyjádření majoritního hlasu, modrý je pak minimální hodnota. Každý žánr obsahoval alespoň jednu skladbu, která byla zařazena jednohlasně, tedy hodnota majoritního hlasu byla 100%.



Obrázek 4.1: Graf zobrazující procentuální hodnotu majoritního hlasu pro jednotlivé žánry

Dle očekávání se někteří lidé u některých testovaných nahrávek neshodli ohledně zařazení. Největší problémy byly v případě rockových, metalových ale i folkových nahrávek, což je vidět i na nižším mediánu majoritních hlasů.

U jazzu se jednalo spíš o problémy s jednotlivými vzorky, s většinou problém nebyl. Na druhou stranu klasická hudba a rap dosáhly u lidí z testovací skupiny téměř jednohlasné klasifikace.

Celkově bylo možné na základě většiny vybrat hudební žánr pro každou z testovaných nahrávek. Pro následující testy jsem používala právě tyto žánry.

4.1.2 Výsledky klasifikátoru

Nyní popíšu výsledky klasifikátoru, kterých dosáhl na testovací množině použité v testu referenční skupiny lidí. Protože jsem během vývoje zkoušela různé přístupy, rozhodla jsem se nechat při spuštění klasifikátoru možnost zvolit typ používaného příznakového vektoru a výsledky proto zobrazuji pro různá nastavení klasifikátoru.

Pro každý příznakový vektor jsem zkoušela oba klasifikační algoritmy, které program poskytuje. Lepších výsledků obecně dosáhla NN, takže dále uvedené výsledky jsou výsledky klasifikace za použití NN.

MFCC

Výsledky klasifikace pouze podle MFCC zobrazuje tabulka 4.1.

klasifikace / žánr	klasika	folk	rap	jazz	rock	metal	%
klasika	8	0	0	1	0	0	$88,\overline{8}$
folk	0	5	0	1	0	0	$83,\overline{3}$
rap	0	1	7	0	0	0	$87,5$
jazz	1	1	0	5	0	0	$71,43$
rock	1	2	0	0	5	2	50
metal	0	0	0	0	1	7	$87,5$

Tabulka 4.1: Výsledky klasifikátoru při použití pouze MFCC jako příznaků

Řádky odpovídají skutečným žánrům, sloupce pak žánrům, do kterých byly vzorky klasifikovány. Sloupec označený symbolem % představuje procentuální úspěšnost zařazení pro daný hudební žánr.

Je vidět, že klasifikace podle MFCC byla poměrně úspěšná pro klasickou hudbu, folk, rap a metal. Naopak výsledky pro rock nejsou příliš přívětivé.

Celkově klasifikátor správně zařadil 37 skladeb ze 48, což odpovídá úspěšnosti 77%. Pro srovnání klasifikace pomocí algoritmu k-NN dosáhla úspěšnosti 70,83%, tj. 34 správně zařazených skladeb.

Spektrální příznaky

Spektrální příznakový vektor dosáhl výsledků zobrazených v tabulce 4.2.

žánr \ klasifikace	klasika	folk	rap	jazz	rock	metal	%
klasika	8	0	0	1	0	0	$88,\overline{8}$
folk	2	4	0	0	0	0	$66,\overline{6}$
rap	0	1	7	0	0	0	$87,5$
jazz	1	1	0	5	0	0	$71,43$
rock	3	1	0	0	6	0	60
metal	0	0	0	0	0	8	100

Tabulka 4.2: Výsledky klasifikátoru pro spektrální příznaky

Celkový počet správně zařazených nahrávek je 38 ze 48, tj. úspěšnost 79%. Tento příznakový vektor byl úspěšný pro metal, elektronickou a klasickou hudbu. I když do klasické hudby bylo zařazeno i značné množství vzorků patřící do jiných hudebních žánrů.

Klasifikace pomocí algoritmu k-NN zařadila správně 36 ze 48 nahrávek, tedy dosáhla úspěšnosti 75%.

Rytmické příznaky

Klasifikace pomocí rytmického příznakového vektoru dopadla dle očekávání nejhůře viz tabulka 4.3.

žánr \ klasifikace	klasika	folk	rap	jazz	rock	metal	%
klasika	8	0	0	0	0	1	$88,\overline{8}$
folk	2	3	0	0	0	1	50
rap	0	1	6	0	0	1	75
jazz	0	1	1	2	1	2	$28,57$
rock	0	1	1	0	4	4	40
metal	0	0	0	0	2	6	75

Tabulka 4.3: Výsledky klasifikátoru pro rytmické příznaky

Celková úspěšnost je 29 správně zařazených skladeb ze 48, tedy lehce přes 60%. Algoritmus k-NN dosáhl pouze 22 správných odpovědí (úspěšnost 45%).

Všechny příznaky

Po přidání rytmického příznakového vektoru ke spektrálnímu se sice zlepšilo špatné klasifikování jiných hudebních žánrů mezi klasickou hudbu, nicméně celková úspěšnost o něco klesla vzhledem k výsledků spektrálního příznakového vektoru.

Tento případ je jediný, kdy lepších výsledků dosáhla klasifikace pomocí k-NN než NN. Výsledky použitím k-NN zobrazuje tabulka 4.4.

klasifikace / žánr	klasika	folk	rap	jazz	rock	metal	%
klasika	7	0	0	2	0	0	$77,\overline{7}$
folk	0	5	0	1	0	0	$83,\overline{3}$
rap	0	0	7	0	1	0	$87,5$
jazz	1	1	0	4	1	0	$57,14$
rock	0	2	1	1	5	1	50
metal	0	0	0	0	0	8	100

Tabulka 4.4: Výsledky klasifikátoru při použití všech příznaků

Celková úspěšnost je 77%, 37 správně zařazených nahrávek. Klasifikace pomocí NN měla celkem 36 ze 48 správných odpovědí, což odpovídá úspěšnosti 75%.

4.2 Vliv parametrů na úspěšnost

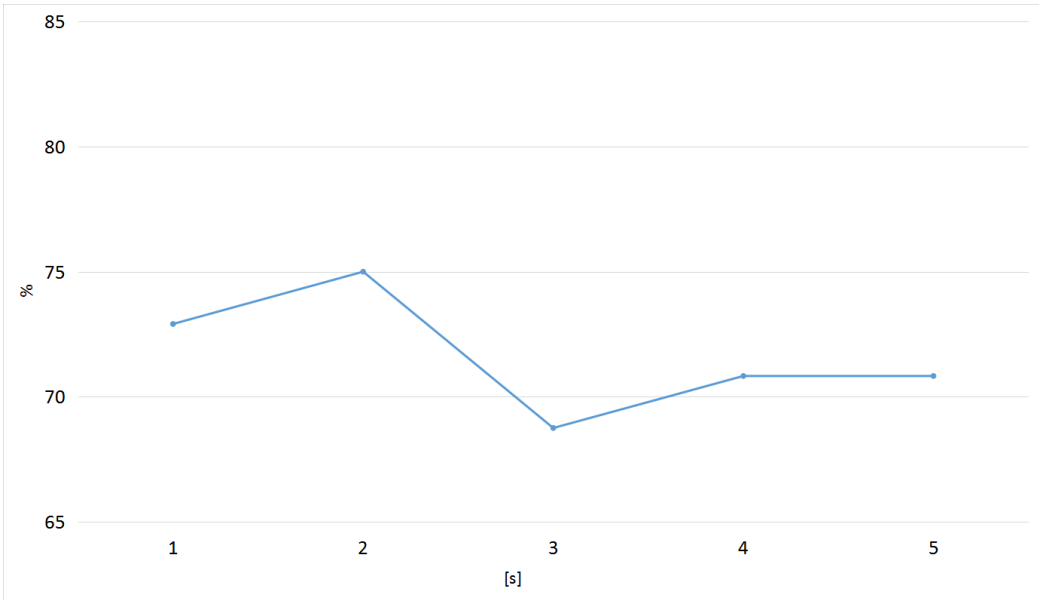
Během realizace jsem zkoušela různé nastavení klasifikátoru. Zde popíši jaký vliv na úspěšnost měly změny jednotlivých parametrů. Zejména se jedná o velikost analytického okna a zpracovávaného vzorku.

Pro následující testy byl použit spektrální příznakový vektor, který dosáhl nejlepších výsledků. Jako klasifikační algoritmus jsem použila k-NN kvůli rychlejší zpětné vazbě. Nebyl problém stejné testy provést i pro NN, ale bylo by to velmi časově náročné. Klasifikace pomocí k-NN poskytuje dobrý odhad úspěšnosti pro jednotlivé parametry.

4.2.1 Velikost zpracovávaného vzorku

Tento parametr určuje pro jak velkou část nahrávky bude klasifikátor určovat žánr. Nahrávka se typicky dělí na více vzorků, ale žánr se určuje pro každý z nich nezávisle.

Během poslechových testů jsem velikost vzorku stanovila na 1 – 2 s. Nicméně, jak je vidět v grafu 4.2, zkoušela jsem i jiné hodnoty a zvolená velikost 2 vteřin skutečně dosáhla nejlepších výsledků, i když jen lehce. Pro delší časové úseky pak úspěšnost mírně klesá.

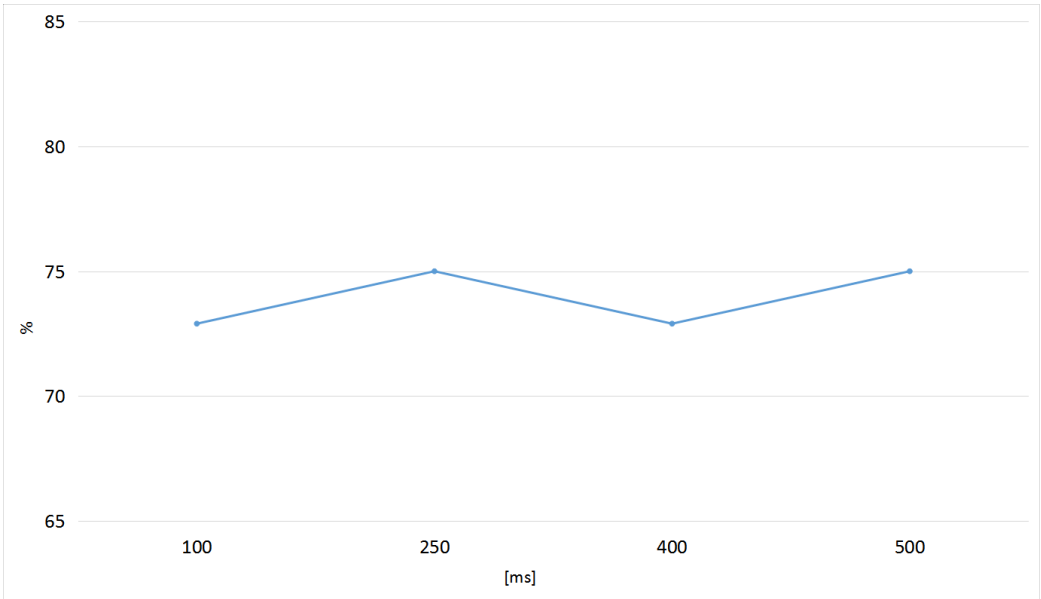


Obrázek 4.2: Graf závislosti úspěšnosti na velikosti vzorku

Pro všechny testy bylo použito nejlepší analytické okno, tedy 250 ms.

4.2.2 Velikost analytického okna

Zpracovávaný vzorek se pro účely extrakce příznaků dále dělí na analytická okna, po kterých je zpracováván. Závislost úspěšnosti klasifikace na velikosti analytických oken zobrazuje graf 4.3.



Obrázek 4.3: Graf závislosti úspěšnosti na velikosti analytického okna

Nejlepších výsledků dosáhla klasifikace při používání 250 ms analytického okna. Pro všechny testy byla použita velikost vzorku 2 s.

4.3 Zhodnocení výsledků

Testu referenční skupiny se účastnilo celkem 23 lidí. Očekávala jsem účast vyšší, ale s nakonec jsem s velikostí skupiny spokojená. Velkou část účastníků nejspíše odradila velikost testovací množiny, která měla něco pod 3 hodiny poslechu. Většina hudebních žánrů získaných z tohoto testu souhlasila s mými původními odhady, několik jsem odhadla špatně a musela jsem jejich žánr upravit.

To navádí k myšlence, že podobně sestavený trénovací dataset by mohl dosáhnout vyšší úspěšnosti než dataset, který jsem připravovala pouze na základě svého osobního názoru.

Celkově dosáhl klasifikátor přiměřeně dobrých výsledků. Porovnávala jsem klasifikaci použitím dvou algoritmů – k-NN a NN, kdy lepších výsledků dosáhla ve většině případů neuronová síť. Dále jsem zkoušela různé přístupy k extrakci příznaků.

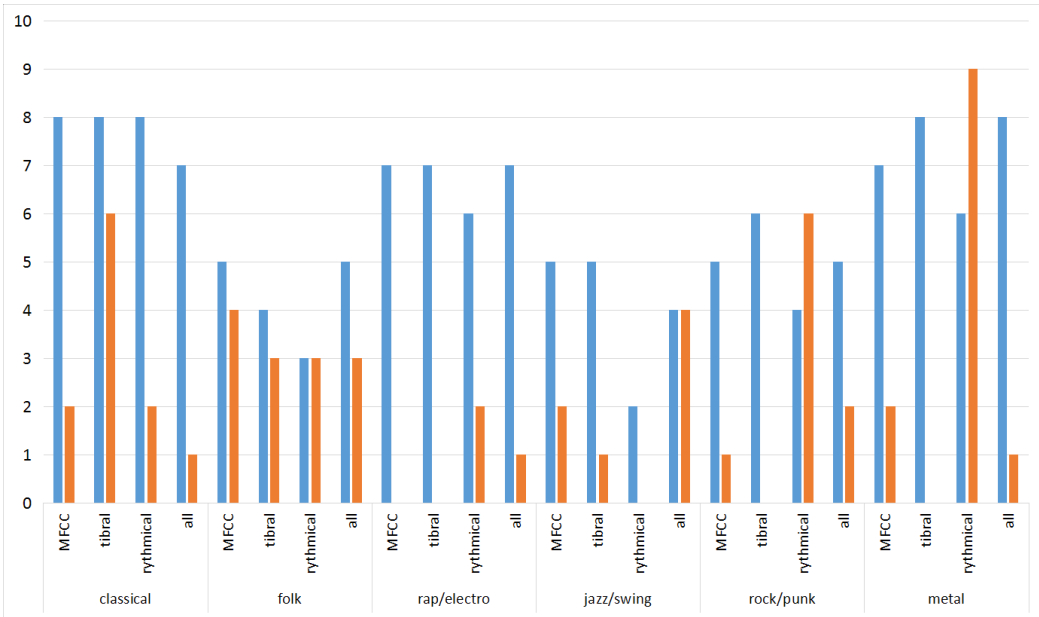
Testovala jsem různá nastavení klasifikátoru a jejich vliv na úspěšnost klasifikace. Zejména jsem zkoušela měnit délku zpracovávaného signálu a délku analytických oken. Pro finální nastavení jsem vybrala ty hodnoty, se kterými byla klasifikace nejúspěšnější – 2 sekundové vzorky rozdělené na 250 ms okénka, která se z poloviny překrývají.

Vytvořila jsem čtyři různé příznakové vektory, z nichž jeden je kombinací všech předchozích. Nejlepších výsledků dosáhl spektrální příznakový vektor, který dokázal určit zařadit správně 38 skladeb ze 48, což odpovídá úspěšnosti 79%.

Naopak nejhůře dopadl rytmický příznakový vektor s úspěšností pouze 60%. Ostatní příznakové vektory dosáhly úspěšnosti kolem 75%.

Klasifikace s pomocí příznakového vektoru má pro jednotlivé skupiny dobrou úspěšnost správných zařazení, velmi často ale klasifikuje chybně skladby jako klasiku, i když by měly patřit do jiné kategorie.

Rytmický příznakový vektor má zase problémy klasifikovat jazz a značné množství skladeb zařazuje chybně do metalu. Poměr správně a chybně zařazených skladeb zobrazuje graf 4.4.



Obrázek 4.4: Poměr správně a chybně zařazených skladeb dle žánrů a příznakových vektorů

Modrá řada značí počet správně zařazených nahrávek, oranžová pak počet skladeb mylně zařazených do daného žánru. Hodnoty je třeba porovnávat v rámci jednoho žánru, každá skupina má totiž různý počet skladeb – klasika 9, folk 6, rap/elektro 7, jazz/swing 8, rock/punk 10 a metal 8. Původně jsem měla v úmyslu vytvořit rovnoměrně zastoupené všechny skupiny, ale výsledky referenčního testu počty skladeb v kategoriích mírně zamíchaly.

5 Závěr

Cílem práce bylo vytvořit klasifikátor, který bude u daných nahrávek ve formátu MP3 klasifikovat hudební žánr. Pro účely práce jsem definovala šest hudebních žánrů, pro které jsem testovala optimální nastavení parametrů klasifikátoru a způsob extrakce příznaků.

Na skupině dobrovolníků s kladným vztahem k hudbě jsem provedla test, díky němuž jsem získala referenční zařazení jednotlivých testovaných nahrávek do skupin hudebních žánrů, se kterým jsem posléze porovnávala výsledky klasifikátoru. Především kvůli celkové délce tohoto testu, bylo dobrovolníků o něco méně, než bylo původně v plánu. Přesto si myslím, že jsou získaná data dostatečně vypovídající.

Program implementuje dva klasifikační algoritmy a používá čtyři různé příznakové vektory, jejichž úspěšnost jsem v závěru práce porovnávala. Nejlepších výsledků dosáhl klasifikátor s použitím spektrálního příznakového vektoru, a to 79%. Rytmický příznakový vektor naopak dopadl nejhůře. Výsledky dalších dvou příznakových vektorů jsou srovnatelné a pohybují se kolem 75%.

Hlavní problém úlohy vidím v trénovací množině dat, kterou jsem sestavovala sama. Ideální by bylo provést obdobný test jako na množině testovací a žánry u vzorových skladeb určit hlasováním ve větším počtu lidí. Vzhledem k množství dat, které bylo třeba pro natrénování klasifikátoru, to bohužel nebylo možné.

Další zlepšení by mohlo být dosaženo výběrem částí skladeb, které odpovídají danému žánru nejlépe. Opět to vede na úpravu trénovací množiny. Nahrávky používané pro trénování se nijak neupravovaly a klasifikátor je používal v původní podobě. Většina nahrávek mohla obsahovat i části typově patřící do jiného hudebního žánru.

V současné době je možné klasifikátor používat na poměrně přesné určení hudebního žánru, i když existuje prostor pro vylepšení jeho úspěšnosti. Program je připraven pro možné další rozšiřování, a to jak přidávání klasifikačních algoritmů, tak změnu hudebních žánrů.

Literatura

[1] Waniata Ryan. *Do touch that dial: How to master your equalizer for the perfect sound* [online]. [cit. 04/2016]. Dostupné z: `http://www.digitaltrends.com/home-theater/eq-explainer/`.

[2] Kostelný. *Frekvence, Panorama a Hloubka.* [online]. [cit. 04/2016]. Dostupné z: `http://www.muzikus.cz/pro-muzikanty-clanky/Frekvence-panorama-a-hloubka-Tema-mesice~21~srpen~2007/`.

[3] Carter Jeremy. *The frequency spectrum, instrument ranges, and EQ tips* [online]. [cit. 04/2016]. Dostupné z: `http://www.digitaltrends.com/home-theater/eq-explainer/`.

[4] Grant. *Music Genres List* [online]. [cit. 04/2016]. Dostupné z: `http://www.musicgenreslist.com/`.

[5] Ambardar Ashok. *Digital Signal Processing. A Modern Introduction.* Thomson Learning, 2006. ISBN 978-0-534-40509-0.

[6] Guaus Enric. *Audio content processing for automatic music genre classification: descriptors, databases, and classifiers.* PhD thesis, Universitat Pompeu Fabra, Barcelona, 2009.

[7] Lyons James. *Practical Cryptography: An Intuitive Discrete Fourier Transform Tutorial* [online]. [cit. 02/2016]. Dostupné z: `http://practicalcryptography.com/miscellaneous/machine-learning/intuitive-guide-discrete-fourier-transform/`.

[8] Lyons James. *Practical Cryptography: Mel Frequency Cepstral Coefficient (MFCC) tutorial* [online]. [cit. 11/2015]. Dostupné z: `http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/`.

[9] Riedmiller Martin, Braun Heinrich. A direct adaptive method for faster backpropagation learning. 1993.

[10] Tzanetakis George, Cook Perry. Musical Genre Classification of Audio Signal. *IEEE Transactions on Speech and Audio Processing.* 2002, 10, 5, s. 293–302.

[11] *FMA – Free Music Archive* [online]. [cit. 11/2015]. Dostupné z: `http://freemusicarchive.org`.

[12] *SoundCloud* [online]. [cit. 11/2015]. Dostupné z: `https://soundcloud.com/`.

[13] Tzanetakis George, Essl George. Automatic Musical Genre Classification Of Audio Signals. 2001, s. 293–302.

[14] *Un4seen Developments – BASS Documentation* [online]. [cit. 10/2015]. Dostupné z: `http://www.un4seen.com/doc/`.

[15] *Fast Artificial Neural Network Library* [online]. [cit. 10/2015]. Dostupné z: `http://leenissen.dk/fann/wp/`.

[16] *libMFCC – C library for computing Mel-Frequency Cepstral Coefficients* [online]. [cit. 10/2015]. Dostupné z: `https://github.com/jsawruk/libmfcc`.

[17] Sturm Bob, Morvidone Marcela, Daudet Laurent. Musical Instrument Identfication Using Multiscale Mel-Frekquency Cepstral Coefficients. 2010.

A Výsledky referečního testu

Tabulky A.1 a A.2 zobrazujíc výsledky testu referenční skupiny dobrovolníků.

track	klasika	folk	elektro/rap	jazz/swing	rock/punk	metal
1	8,33	**91,67**	-	-	-	-
2	-	**100**	-	-	-	-
3	-	-	-	-	33,33	**66,67**
4	-	-	-	**100**	-	-
5	-	-	**100**	-	-	-
6	-	-	**95,83**	-	4,17	-
7	-	-	-	-	-	**100**
8	**100**	-	-	-	-	-
9	-	-	**100**	-	-	-
10	-	-	-	**100**	-	-
11	-	-	**100**	-	-	-
12	**100**	-	-	-	-	-
13	-	-	-	**100**	-	-
14	-	-	-	-	45,83	**54,17**
15	-	-	-	**100**	-	-
16	-	-	4,17	-	25,00	**70,83**
17	-	29,17	-	**70,83**	-	-
18	**79,17**	8,33	12,50	-	-	-
19	-	**83,33**	-	4,17	12,50	-
20	-	-	-	-	**100**	-
21	-	-	-	**100**	-	-
22	**100**	-	-	-	-	-
23	**100**	-	-	-	-	-

Tabulka A.1: Výsledky testu referenční skupiny – část 1

Každý řádek odpovídá jedné skladbě v testovací množině – byly označeny *track* a pořadové číslo.

Hodnoty na řádcích odpovídají procentuálnímu vyjádření hlasů lidí pro daný žánr a skladbu. Např. je vidět, že nahrávku označenou jako *track01* zařadilo 8,33% lidí do skupiny klasická hudba a 91,67% lidí ji zařadilo fo folku.

Zvýrazněné hodnoty jsou majoritní hlas skupiny – a také žánr, který byl dané testovací nahrávce po tomto testu přiřazen.

track	klasika	folk	elektro/rap	jazz/swing	rock/punk	metal
24	-	**100**	-	-	-	-
25	-	-	-	-	**62,50**	37,50
26	-	-	8,33	-	41,67	**50**
27	-	-	4,17	-	**95,83**	-
28	-	8,33	12,50	-	**79,17**	-
29	-	-	-	-	-	**100**
30	-	-	8,33	-	16,67	**75**
31	-	-	-	**100**	-	-
32	-	-	**100**	4,17	-	-
33	**95,83**	4,17	-	-	-	-
34	-	**100**	-	-	-	-
35	-	4,17	-	-	**91,67**	4,17
36	-	**91,67**	-	4,17	4,17	-
37	-	-	**100**	-	-	-
38	**95,83**	-	-	-	-	-
39	-	16,67	16,67	16,67	**50**	-
40	-	-	**100**	-	-	-
41	-	29,17	-	12,50	**58,33**	-
42	-	4,17	4,17	-	**91,67**	-
43	-	4,17	**95,83**	-	-	-
44	**100**	-	-	-	-	-
45	-	37,50	-	-	**62,50**	-
46	**100**	-	-	-	-	-
47	-	12,50	8,33	4,17	**75**	-
48	-	-	-	-	4,17	**95,83**

Tabulka A.2: Výsledky testu referenční skupiny – část 2

Z tabulek jsou vidět dva největší problémy dobrovolníků. Často se nemohli rozhodnout mezi rockem a metalem, viz nahrávky číslo 14 a 26. Druhý problém byl u zařazení skladeb, které dle výsledků pravděpodobně patří do více kategorií – zde se jedná např. o nahrávky číslo 39 a 41.

B Uživatelská příručka

Program `MusicGenreClassifier` je konzolová aplikace pro klasifikaci záznamů hudby ve formátu MP3 do skupin dle hudebních žánrů. Skupin hudebních žánrů, které klasifikátor umí určit, je šest, a to

1. klasická hudba,
2. folk,
3. elektronická hudba a rap,
4. jazz a swing,
5. punk, rock a
6. metal.

B.1 Souborová struktura

Souborová struktura na přiloženém CD s programem je následující:

- složka `code` obsahuje zdrojové soubory programu,
- složka `project` obsahuje zdrojové soubory a projekt pro Microsoft ® Visual Studio ®,
- složka `bin` obsahuje přeložený spustitelný soubor pro OS Microsoft ® Windows ®.

B.2 Instalace programu

Pro spuštění souboru stačí nakopírovat složku `bin` na svůj pevný disk a pokračovat dle návodu pro práci s programem B.3.

Pro přeložení programu doporučuji použít Microsoft ® Visual Studio ® Community 2015 [1], ve kterém lze otevřít soubor `MusicGenreClasifier.sln`, nacházející se ve složce `visual_studio_files`. Projekt má nastavené cesty k používaným knihovnám a nic není třeba upravovat.

Překlad spustíte příkazem `Build` > `Build Solution`. Přeložený spustitelný soubor lze poté najít ve složce `Release`, resp. `Debug`, ve stejné složce jako soubor projektu.

Pro spuštění programu je třeba ve stejné složce mít dynamické knihovny `fanfloat.dll` a `bass.dll`, které lze zkopírovat ze složky `bin`.

B.3 Práce s programem

Klasifikátor lze spustit ve 3 různých módech – příprava dat pro trénování, trénování klasifikačního algoritmu a klasifikace. Mód klasifikátoru, případně další nastavení, se ovládá pomocí parametrů zadaných při spuštění.

-F path slouží pro zapnutí módu extrakce příznaků, za parametrem musí následovat cesta ke kořenovému adresáři s trénovacími daty.

-T zapne mód trénování.

-R path zapne mód klasifikace, za parametrem musí následovat cesta ke složce obsahující skladby pro klasifikaci.

Parametry mohou být použity každý zvlášť nebo se mohou kombinovat. V případě chybného zadání parametrů vypíše program nápovědu a skončí.

B.3.1 Extrakce příznaků

Pro extrakci příznaků je třeba za parametr přidat cestu k adresáři s trénovacími daty. Pro správné načtení všech dat musí být dodržena přesně daná struktura. Kořenový adresář obsahuje 6 složek označených `01` – `06`, které odpovídají jednotlivým hudebním žánrům.

[1] Zdarma dostupné na `https://www.visualstudio.com/`

1. Složka `01` odpovídá klasické hudbě,
2. složka `02` folku,
3. složka `03` je určena pro elektronickou hudbu a rap,
4. složka `04` pro jazz a swing,
5. složka `05` bude obsahovat punk a rock a
6. složka `06` metal.

V těchto složkách jsou umístěné trénovací nahrávky rozdělené dle daných žánrů.

Výstupem extrakce příznaků je soubor `input.data`, který se vytvoří ve složce, odkud byl klasifikátor spuštěn. Tento soubor obsahuje příznaky pro trénovací množinu a je nutný pro spuštění trénování.

B.3.2 Trénování

Pro trénování je třeba, aby ve složce s programem byl soubor `input.data`. Poté dojde k trénování neuronové sítě nad připravenými daty. Výstupem trénování je soubor `amgc.net`, který obsahuje uloženou neuronovou síť a používá se pro další krok – klasifikaci.

B.3.3 Klasifikace

Klasifikace vyžaduje přítomnost souboru `amgc.net` se strukturou neuronové sítě. Za parametr `R` je třeba přidat cestu k adresáři s testovanými nahrávkami. Klasifikace probíhá v současné verzi programu dávkově, proto nelze zadat pouze jeden soubor, ale musí být uvedena cesta k nadřazené složce. Zpracovány budou všechny zvukové soubory typu MP3. Výstup klasifikace zobrazuje B.1.

Po spuštění vypíše program informativní hlášení o nastavení klasifikátoru a spustí se klasifikace pro jednotlivé soubory v zadané složce. Pro každý MP3 soubor se provede klasifikace a vypíše řádek se zvoleným žánrem a procentuální zastoupení vzorků, které danému žánru odpovídaly.

```
INFO: Selected parameters:
Analyse window size: 0.250000 s
Texture window size 2.000000 s
Number of used MFCC : 20
Number of features(computed) : 24
Used features : MFCC, spectral centroid and spectral rolloff means and
    variances
Selected algorithm : NN
Running test on dir ../test/big/...

Result for the sample '../test/big//track02.mp3':
  1 - folk (75.45 %)

Result for the sample '../test/big//track03.mp3':
  5 - metal (77.72 %)
```

Kód B.1: Příklad výstupu klasifikace

B.3.4 Nepovinná nastavení klasifikátoru

Nepovinné parametry mění nastavení klasifikátoru. Jejich úprava může mít negativní vliv na úspěšnost klasifikace či délku trénování. Standardně program používá nastavení, které v testech dopadlo nejlépe.

Parametry, které je možné upravovat, jsou velikost zpracovávaného vzorku, velikost analytického okna, použitý příznakový vektor a počet použitých Mel-frekvenčních kepstrálních koeficientů (dále MFCC).

Velikost zpracovávaného vzorku na 1 sekundu se změní přidáním parametru `-t=1`. Pro změnu velikosti analytického okna na 200 ms zadejte `-a=200`. Počet MFCC na 12 nastavíte zadáním `-m=12`.

Použitý příznakový vektor lze změnit pomocí parametru `f`. Hodnoty parametru pro jednotlivé příznakové vektory jsou

- `-f=0` pro používání pouze MFCC,
- `-f=1` pro spektrální příznakový vektor,
- `-f=2` pro rytmický příznakový vektor a
- `-f=3` pro kombinovaný příznakový vektor.