MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Marek Čermák

Automatické rozpoznání hudební notace ze zvukových dat

Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. RNDr. Jakub Lokoč, Ph.D.

Studijní program: Informatika

Studijní obor: Programování a softwarové systémy

Praha 2020

Rád bych poděkoval svému vedoucímu práce, doc. RNDr. Jakubu Lokočovi, Ph.D., za veškerou pomoc a trpělivost poskytnutou při psaní tohoto textu a za všechny připomínky a rady, které nemalou měrou přispěly k jeho finální podobě.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V …..... dne..... podpis

Název práce: Automatické rozpoznávání hudební notace ze zvukových dat

Autor: Marek Čermák

Katedra / Ústav: Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. RNDr. Jakub Lokoč, Ph.D.

Abstrakt: Cílem této práce je návrh a implementace aplikace využívající konvoluční neuronové sítě k tvorbě hudební notace ze zvukových dat. Aplikace je schopná učit neuronovou síť pomocí vstupních souborů ve formátu MIDI (Musical Instrument Digital Interface) a spárovat jednotlivé úseky hudby s jejich zvukovou podobou. Učení neuronové sítě může probíhat na uživatelem specifikované kolekci souborů MIDI či na náhodně generované hudbě. Každému nástroji ve standardu MIDI může být přiřazena síť, jejímž výstupem jsou přehrávané noty v daném časovém úseku. Postupným procházením zvukových dat generuje síť úseky aktivních not, které jsou následně spojeny do výsledného souboru. Součástí aplikace je také rozpoznávání slov ze zvuku pomocí externí služby.

Klíčová slova: hudební notace, neuronová síť, hluboké učení, rozpoznání zvuku, MIDI

Title: Automatic recognition of musical notation from audio data

Author: Marek Čermák

Department: Department of Software Engineering

Supervisor: doc. RNDr. Jakub Lokoč, Ph.D.

Abstract: The goal of this thesis is the design and implementation of an application using convolutional neural networks to generate musical notation from audio data. The application is able to train a neural network using input files in the MIDI (Musical Instrument Digital Interface) format and pair all sections of the music with their audio form. The training of the neural network can be performed on a user-specified collection of MIDI files or on randomly generated music. Each instrument in the MIDI standard can be assigned a network whose output are the notes playing in the given time section. Continuously iterating over the audio data, the network generates sections of active notes which are then concatenated into the output file. The application is also capable of recognizing words from audio using an external service.

Keywords: musical notation, neural network, deep learning, audio recognition, MIDI

Obsah

1. Úvod

Hudba představuje nedílnou část lidské kultury. Od svého původního rituálního významu se postupně rozšířila jako součást umění do mnoha oblastí celého světa a s rozvojem moderní techniky se její poslouchání stalo pro nespočet lidí každodenní činností.

Hudba sama o sobě se dá nejobecněji popsat jako velmi komplexní spojení zvuků rozličných hudebních nástrojů. Tato práce si klade za cíl toto spojení (ve formě zvukové vlny) analyzovat a převést do podoby, se kterou se dá z hlediska struktury hudby pracovat lépe než se zvukovou vlnou, a sice do podoby hudební notace.

1.1 Záznam hudby

V mnoha lidských kulturách již od počátku vývoje hudby vyvstala nutnost mít nějaký způsob, jak zaznamenat hudbu pro její pozdější reprodukci. Z tohoto důvodu začaly vznikat různé hudební notace schopné zaznamenat výšku tónu, nástroj, hlasitost, tempo a jiné parametry hudby, často společně i s textem.

První systémy hudební notace v evropských kulturách představovaly tzv. *neumy*, rozličné značky nad slovy udávající výšku v daném místě písně. První pokročilejší systém hudební notace tohoto typu vznikl v Byzantské říši. Od dnes nejrozšířenějšího typu hudební notace se tento systém lišil především tím, že značky nad slovy udávaly výšku tónu nikoliv absolutně, ale relativně vůči předchozímu tónu, a tak při reprodukci hudby zapsané v této notaci bylo potřeba vědět, jaký je základní tón a jak velkou změnu výšky představují značky pro vyšší či nižší tón.

Moderní hudební notace (vzniklá v době renesance) je založena na umisťování jednotlivých not (představující oddělené tóny) do notové osnovy, přičemž vertikální pozice noty určuje její výšku a horizontální pozice noty určuje začátek tónu v rámci taktu. Dalšími popisky jsou určeny jiné parametry hudby, například tempo (doba jednoho taktu) či hlasitost.

Díky moderním způsobům záznamu zvuku je v současnosti nejběžnějším způsobem reprezentace hudby záznam zvukové vlny vzniklé při reprodukci hudby, která přesně reprezentuje zvuky všech nástrojů v dané skladbě. Zvukovou vlnou je v tomto případě myšlena funkce okamžité výchylky mechanického oscilátoru, kterým se říší zvukové vlnění, v závislosti na čase. Příkladem mechanického oscilátoru může být membrána reproduktoru, která svým kmitáním způsobuje změnu tlaku okolního média a tedy vydává zvukové vlnění. Membrány lidského ucha jsou též mechanické oscilátory, které naopak zvukové vlnění přijímají a stojí na počátku mechanismu lidského smyslu sluchu (Lepil, 2009). Záznam vibrace mechanického oscilátoru tedy zachycuje mechanickou podstatu tvorby, šíření i přijímání zvuku.

Záznam hudby v této podobě lze realizovat analogově, tedy na médium podporující spojitý záznam (gramofonové desky, magnetofonové pásky), nebo dnes nejčastěji digitálně na jakékoliv médium určené pro záznam digitálních dat. V obou případech je důležitým parametrem při záznamu amplituda původní zvukové vlny, tedy maximální absolutní hodnota výchylky mechanického oscilátoru (Lepil, 2009), jehož kmitání je zaznamenáváno. Při záznamu je hodnota okamžité výchylky převedena z původního rozsahu, daného amplitudou, do rozsahu hodnot podporovaného záznamovým médiem. Tento rozsah standardně bývá v teorii $[-1; 1]$,

ale reálná záznamová média mají jiný rozsah (např. digitální záznam pomocí 16bitových znaménkových vzorků má rozsah hodnot [-32768; 32767][1]). Dojde-li k pokusu o záznam výchylky mimo očekávaný rozsah vstupních hodnot, musí být záznamové zařízení připraveno reagovat na tuto situaci dočasnou nebo trvalou změnou rozsahu následujících nebo i předešlých zaznamenaných hodnot. Pokud se tak nestane, dojde k omezení zaznamenaných hodnot do povoleného rozsahu, a tedy k ořezu zvuku, což má na zvuk zkreslující efekt. K ořezu může též docházet při přílišném zvýšení hlasitosti, které má za následek zvětšení některých výchylek mimo povolený rozsah[2].

1.2 Digitální záznam zvuku

Praktická digitální reprezentace zvuku vyžaduje diskretizaci zvukové vlny a použití vzorkování, tedy rozdělení zvukové funkce na úseky pevné délky a přiřazení jedné hodnoty každému úseku (z hodnot funkce v daném úseku). Tato posloupnost vzorků poté reprezentuje původní zvukovou funkci.

Při ukládání vzorků hraje roli i formát ukládání hodnot jednotlivých vzorků. Jelikož nelze digitálně v konečné paměti reprezentovat libovolné reálné číslo, je při ukládání hudby potřeba také specifikovat číselný typ použitý pro hodnoty vzorku. To bývají nejčastěji znaménková 16bitová celá čísla v rozsahu od −32768 do 32767, na nějž je mapován původní rozsah hodnot. Jeden vzorek zvukové vlny v tomto formátu zabírá dva bajty v digitálním datovém proudu.

Důležitým parametrem pro reprezentaci zvuku je také vzorkovací frekvence, jež udává počet vzorků obsažených v jedné sekundě zvukové vlny. Je-li frekvence příliš nízká, nebude zvuk zcela dokonale reprezentován, naopak je-li frekvence příliš vysoká, digitální reprezentace zvuku bude zabírat příliš mnoho místa.

Vhodnou vzorkovací frekvenci pro digitální záznam zvuku složeného ze sinových vln udává Nyquistův-Shannonův teorém (Shannon, 1949):

$$f_v > 2f_{max}$$

Je-li největší frekvence sinových vln f_{max}, dvojnásobek této frekvence stačí jako vzorkovací frekvence f_v. V praxi se používá nejčastěji hodnota 44,1 kHz, neboť lidské ucho není schopno zaznamenat frekvence v průměru nad 20 kHz.

Tento způsob záznamu je schopen věrně reprezentovat jakýkoliv zvuk (s frekvencemi v daném konečném intervalu), ale použití pouze tohoto způsobu má mnohé nevýhody. Prvním problémem je velikost, neboť při standardních

---

[1] Počet přirozených čísel v tomto rozsahu je 2^{16} , tedy počet různých hodnot reprezentovatelných 16bitovým číselným typem.

[2] Jednoduchý nahrávací software může, dojde-li ke zvýšení hlasitosti zvuku nad očekávanou hodnotu, způsobit ořez a tedy zvuk zkreslit se ztrátou informace v místě ořezu. Pokročilejší nahrávací software může v témže případě znovu normalizovat celý záznam tak, aby odpovídal nové amplitudě celé vstupní vlny, nebo může dynamicky upravit zaznamenávaný rozsah hodnot tak, aby odpovídal amplitudě jen části zvukové vlny v okolí zaznamenávané výchylky (dynamická normalizace).

parametrech by hodina záznamu zabírala něco přes 300 MiB[3]. Druhým problémem je obtížnost úprav takto uložené hudby. Jelikož se jedná o záznam zvukové vlny složený ze všech nástrojů hrajících přes sebe, je strojově netriviální izolovat jednotlivé nástroje a upravovat jejich zvuk. Úpravy takto uložené hudby jsou omezeny pouze na změnu či omezení frekvence, tempa či hlasitosti celé vlny či jejích úseků.

1.3 Hudební formáty

V současnosti je téměř vždy použit pro uložení hudby některý z nespočtu formátů. Kromě umožnění zaznamenání metadat (například právě vzorkovací frekvence či velikosti vzorků) nabízí většina formátů taktéž řešení jednoho z problémů digitálního ukládání zvuku (nebo obou).

Formáty použitelné pro záznam jakéhokoliv zvuku (nejen hudby) nejčastěji přidávají navíc kompresi, která podstatně snižuje velikost výsledných dat. Taková komprese funguje nejčastěji na bázi izolování jednotlivých frekvencí či odstranění redundantních dat. Tyto formáty se dají dále dělit na ztrátové (například MP3, Ogg/Vorbis nebo AAC), jejichž komprese způsobí ztrátu jisté informace z původních dat a nemožnost zcela přesné reprodukce původní zvukové vlny (která je však stále lidským uchem nerozpoznatelná), a bezztrátové (například FLAC), které za cenu větší velikosti komprimovaných dat dokážou rekonstruovat původní zvuk bez jakýchkoliv chyb.

Druhou velkou skupinu formátů pro záznam hudby představují formáty založené nikoliv na zvukové vlně, ale na její struktuře z pohledu hudební teorie, izolující jednotlivé nástroje, tóny, akordy a jejich parametry jako hlasitost, tempo, výška apod. Přestože s rozvojem techniky záznamu zvuku bylo umožněno ukládat hudbu jako zvukovou vlnu, stále se často využívá tento odlišný způsob záznamu hudby, ve své podstatě podobný klasické historické hudební notaci, neboť nabízí mnohem snazší možnost prohlížení či úpravy parametrů hudby a také obecně stále zabírá mnohem méně místa než komprimovaná zvuková vlna. Na data uložená v těchto formátech lze nahlížet deklarativně (jako na popis složení hudební skladby) nebo imperativně (jako na instrukce pro přehrávací zařízení). Hardwarově i softwarově řešené nástroje pro přehrávání takto uložené hudby se nazývají syntetizéry.

Z charakteru těchto hudebních formátů vychází, že ne všechny lze použít pro záznam hudby odpovídající libovolné zvukové vlně, ale jen takové hudby, kterou dokáže reprodukovat příslušný syntetizér. Z toho důvodu je většina těchto hudebních formátů vázána na konkrétní typy syntetizérů. Například pro čip Yamaha YM3812 (známý jako OPL2), používaný ve zvukových kartách AdLib, existují formáty jako IMF (proprietární formát společnosti id Software), DRO (DOSBox Raw OPL, používaný při nahrávání hudby v nástroji DOSBox), ROL, MUS či MDI[4] (všechny od společnosti Ad Lib) a nespočet dalších méně používaných či proprietárních

---

[3] $3600\ \text{s} \cdot 44100\ \text{Hz} \cdot 16\ \text{bit} = 2540160000\ \text{bit} = 317520000\ \text{B} \cong 302\ \text{MiB}$

[4] Formát MDI je zajímavý tím, že se strukturou shoduje s níže podrobněji popsaným formátem MIDI, ale jeho příkazy jsou určené pro OPL2-kompatibilní zařízení a nikoliv MIDI-kompatibilní. Přesto však lze občas takto uloženou hudbu přehrát jako MIDI, ale výsledek nezní správně.

formátů. Čip Yamaha YM3812 a jemu podobné využívají FM syntézu (frekvenční modulaci) založenou na modifikaci základní vlny (nejčastěji sinus, trojúhelník nebo čtverec) a umožňují takto vlny i skládat (aditivně nebo pomocí modulace), nastavovat parametry jako hlasitost či výšku či přidávat efekty jako tremolo nebo vibrato. Pozdější formáty taktéž umožnily i reálně znějící nástroje, buď s využitím velké databáze vzorků zvuků nástrojů (tzv. soundfontu) jako MIDI, nebo zahrnutím vzorků přímo do souboru s hudbou (formáty MOD, XM či IT)[5] (Category:Formats).

1.4 Standard MIDI

Obrovské množství proprietárních (vzájemně často nekompatibilních) syntetizovaných hudebních formátů vedlo k nutnosti zavést nové standardy pro jednotný způsob reprezentace hudby i nástrojů.

Jedním z těchto standardů je MIDI (Musical Instrument Digital Interface), spravovaný organizací MIDI Manufacturers Association. Tento standard definuje způsoby ukládání a transportu not a dalších parametrů použitých při reprodukci hudby, např. zvolené nástroje, hlasitost, prostorovost apod. Jako tomu je u podobných způsobů ukládání syntetizované hudby, tak ani v tomto standardu nelze uložit přímo zvuk produkovaný nástroji a místo toho musí každé kompatibilní zařízení být schopno reprodukovat zvuky všech nástrojů definovaných standardem podle parametrů specifikovaných v hudebních datech (MIDI Manufacturers Association).

Hudební data jsou uložena jako posloupnost příkazů pro virtuální zařízení. Příkazy mohou být systémové (např. zobrazení informací o skladbě, autorovi apod. nebo nastavení tempa) či konkrétní pro daný zvukový kanál.

Standard MIDI povoluje nastavení parametrů not v nejvýše 16 kanálech. Základní příkazy jsou *Note On* a *Note Off*, které určují, jakou notu má kanál začít či přestat hrát. Mezi další příkazy patří *Control Change* (změna ovladačů nástroje, např. různých pedálů) nebo *Channel Pressure* (změna tlaku na klávesy).

Výška tónu je ve standardu MIDI určena pomocí čísla noty, nikoliv pomocí frekvence. Rozložení not odpovídá chromatické stupnici, která oktávu rozděluje na 12 tónů (C, C♯/D♭, D, D♯/E♭, E, F, F♯/G♭, G, G♯/A♭, A, A♯/H♭, H[6]). Základní tón (A4, 440 Hz) má v MIDI přiřazeno číslo 69[7]. Jelikož platí, že zvýšení noty o jednu oktávu zvýší frekvenci dvojnásobně, následující nota má vždy frekvenci násobenou dvanáctou odmocninou ze dvou. Po notě A4 následuje nota A♯4 (či H♭4), jejíž číslo

---

[5] Tyto formáty nabízejí jistý kompromis mezi čistě syntetizovanou hudbou a reálně znějící hudbou. Aranžér (člověk připravující hudbu pro dané zařízení či formát) může vyhledat reálně znějící zvuky nástrojů použitých ve skladbě a pouze na ně ve skladbě odkazovat, takže výsledná zvuková vlna vznikne jen jako složení opakovaného přehrávání téhož zvuku (kterému lze samozřejmě upravovat hlasitost, výšku či jiné parametry). Technicky tak lze uložit *celou* zvukovou vlnu hudby do takového formátu, ale nebudou tak naplno využity jeho funkce a výsledek bude použitelný jen pro přehrávání.

[6] Nota H je v angloamerických zdrojích označována jako B.

[7] Některé materiály začínají při číslování od 0, jiné od 1. V této práci je použito číslování počínající 0 u všech parametrů.

je 70 a přibližná frekvence 466,16 Hz. Vztah mezi číslem noty a frekvencí je tedy následující:

$$n = 69 + 12 \log_2 \frac{f}{440\text{Hz}}$$
$$f = 2^{\frac{n-69}{12}} \cdot 440\text{Hz}$$

Formát zprávy o změně noty dovoluje 128 možných not. Nejnižší nota s číslem 0 odpovídá základní přibližné frekvenci 8,18 Hz, zatímco nejvyšší nota s číslem 127 odpovídá frekvenci 13289,75 Hz

Nástroj (v MIDI nazývaný „program“) lze nastavit každému kanálu pomocí zprávy *Program Change*. Formát této zprávy umožňuje výběr z nejvýše 128 nástrojů. Základní standard MIDI nespecifikuje přesné nástroje ani jejich mapování na konkrétní čísla, ale nejčastěji používaný standard, který toto udává, je General MIDI, jenž je využit i v této aplikaci. General MIDI rozděluje nástroje do několika kategorií po 8 nástrojích. Kategorie jsou (v tomto pořadí): piano, chromatické bicí (např. xylofon), varhany, kytara, basa, strunné nástroje, ansámbl (hlas, orchestr), žesťové nástroje, dřevěné nástroje, dechové nástroje, syntetické vlny, syntetické akordy, efekty (déšť, ozvěna), exotické nástroje, bicí a zvukové efekty (dech, příboj, ptáci, výstřel apod.).

Bicí nástroje, které nemají měnitelnou výšku, jsou v General MIDI reprezentované pomocí kanálu s číslem 9. Příkazy *Note On* a *Note Off* namísto ovládání tónů přehrávají různé typy bicích nástrojů (buben, tleskání, tom-tom, činel, zvon, triangl apod.). General MIDI definuje 47 typů bicích nástrojů.

Pozdější standard General MIDI 2 umožňuje až 256 nástrojů, jejichž přehrávání je dosaženo pomocí zprávy *Bank Select*, a také rozšiřuje rozsah bicích nástrojů o několik bicích souprav, které se dají měnit příkazem *Program Change*.

Konkrétní zvuky či kvalita nástrojů nejsou nijak standardem podchyceny a závisejí pouze na konkrétním syntetizéru. Některé nástroje umožňují sadu zvuků nahradit jinou a simulovat tak různá zařízení. Takové zvukové „paletě“ se zpravidla říká *soundfont*.

Ačkoliv byl původně standard MIDI navržen primárně pro komunikaci mezi hudebními zařízeními, jeho využití je v současné době převážně ukládání hudebních dat do souborů. Proto je potřeba také stanovit formát souboru, který má obsahovat zprávy MIDI. Nejčastějším formátem je *Standard MIDI File* (SMF, .mid). Tyto soubory jsou rozděleny do několika stop, které mohou být přehrávány za sebou i přes sebe (jedná se pouze o logické rozdělení; kanály jsou sdílené napříč stopami). SMF podporuje také metadata ve formě názvu skladby, stop, autora, textu písně či jiných poznámek.

V záhlaví souboru MIDI je také uloženo časování souboru, tedy kolik taktů interních hodin rozděluje jeden úder (který obvykle odpovídá jedné notě čtvrťové). Všechny časové prodlevy v souboru jsou totiž uloženy v taktech, nikoli ve zlomcích sekund, aby se dala jednoduše upravovat rychlost skladby. Aby mohl být soubor MIDI správně přehrán, je potřeba znát ještě jeho tempo, které se nastavuje ve zprávě *Set Tempo*, nastavující dobu trvání noty čtvrťové v mikrosekundách. Zpráva *Set Tempo* může být poslána i vícekrát.

Často používané časování je 120 taktů za úder a výchozí tempo je 500000 μm za úder. Při těchto hodnotách je doba trvání jednoho taktu, tedy nejmenší

reprezentovatelný časový interval $\frac{500000\ \mu m}{120} = 4{,}1\bar{6}\ \text{ms}$. Podíl tempa a časování hudby nazveme rozlišením hudby.

1.5 Přínos práce

Tato práce si stanovuje dva hlavní cíle. Prvním je vytvoření uživatelsky přívětivé aplikace, kterou může běžný uživatel ovládat a používat podle svých vlastních potřeb jak pro trénování neuronové sítě vytvořené na míru dané hudbě, tak pro aplikování této sítě na hudební data a tvorbu hudební notace.

Druhým cílem je ukázat možnost trénování takovéto sítě na automaticky vygenerovaných datech. Vytvořená aplikace dovede vygenerovat náhodnou hudbu ve formátu MIDI s danými parametry, již je možné následně použít pro trénování sítě pro rozpoznání hudby s podobnými parametry.

1.6 Struktura práce

V následující kapitole 2. Související práce jsou uvedeny některé podobné a související práce. Kapitola 3. Metodika rozpoznávání hudby popisuje celý proces rozpoznávání hudby, od přípravy zvukové stopy po fázi trénování sítě, a zároveň vysvětluje některé související problémy, jako například rozlišování více nástrojů zároveň či polyfonie. Kapitola 4. Ovládání a vývoj aplikace představuje uživatelskou a vývojářskou dokumentaci aplikace; obsahuje popis ovládání, vysvětlení uživatelského rozhraní, strukturu kódu a klíčové komponenty a v kapitole 5. Evaluace jsou uvedeny ukázky chování aplikace při různých modelech neuronových sítí, parametrech rozpoznávání a vstupních dat.

2. Související práce

Samotné problematice rozpoznávání hudby se již věnovali Osmalskyj a kol. (2012), kteří navrhují model rozpoznávání akordů pomocí dopředných neuronových sítí. Pro tvorbu vstupu sítě využívají metodu extrakce profilu výšky (jejíž součástí je i tvorba spektrogramu a aplikování dalších filtrů na něj). Pro tvorbu dostatečných trénovacích dat si autoři práce museli vytvořit vlastní databázi 2000 různých nahrávek akordů na kytaře. Jejich metoda je dostatečně efektivní pro rozpoznávání jednotlivých akordů, ale neumožňuje rozpoznávání libovolných nástrojů a konkrétních tónů (pouze omezeného množství nejčastějších akordů).

Nejnovějším metodám rozpoznání melodie se věnuje Jiří Balhar (2019), který ve své práci podrobně popisuje různé přístupy k rozpoznávání melodie, charakteristiku hudby a roli melodie v ní. Představuje a analyzuje také tři architektury použitelné pro extrakci melodie: architekturu CREPE (Kim a kol., 2018), použité pro sledování jednohlasu, architekturu WaveNet (van den Oord a kol., 2016), která je primárně navržená na generování zvukového signálu, a vlastnoručně navrženou novou architekturu *Harmonic Convolutional Neural Network*, která rozšiřuje standardní konvoluci o přidání harmonické složky signálu do vstupu vrstev.

Velmi podobné metody a postupy udává Verma (2017) při popisu aplikace pro rozpoznávání hudby, na jejímž vývoji se podílel. V době vytváření aplikace popsané touto prací mi nebyl jeho článek znám, ale přesto jsem přistoupil k řešení tohoto problému stejným způsobem, tedy vytvořením spektrogramu ze zvukové vlny, jeho rozdělením na jednotlivé časové úseky pro konvoluční neuronovou síť a následné spojení výstupu sítě. Také v použití formátu MIDI se aplikace shoduje s tou mojí, ale já používám pohyblivé okno nad spektrogramem (s časovým posunem $4{,}1\overline{6}$ ms na rozdíl od 125 ms a s nastavitelnou velikostí kontextu) a moje aplikace navíc obsahuje i možnost generování vlastní hudby a nevychází pouze z dostupných hudebních databází pro učení sítě.

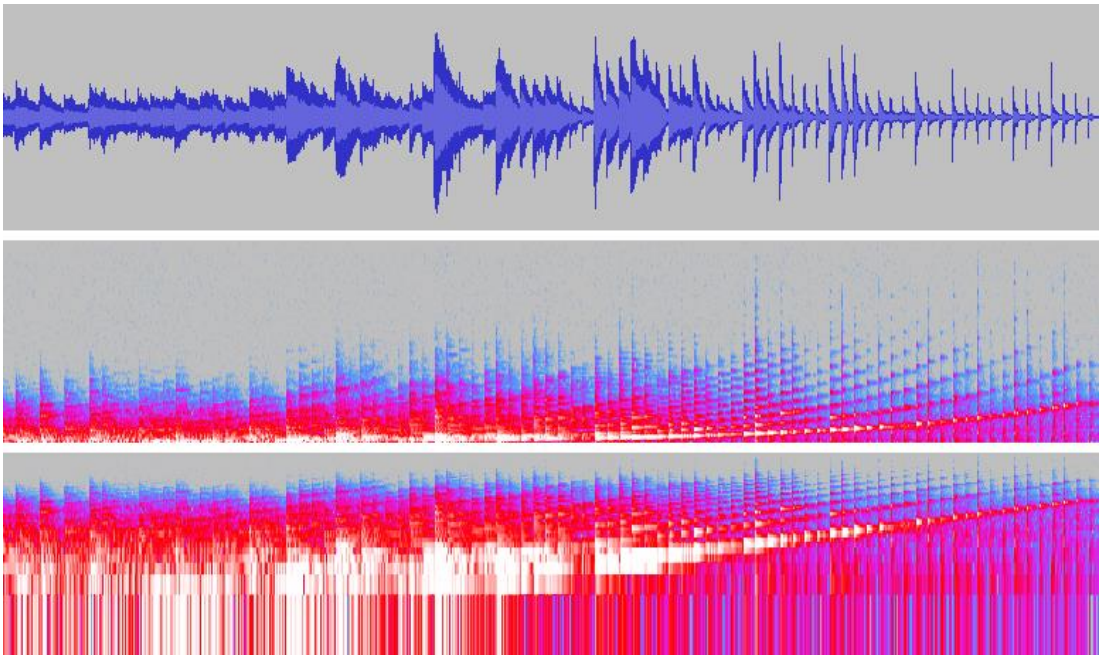
3. Metodika rozpoznávání hudby

Pro účely rozpoznávání hudební notace využívá aplikace konvoluční neuronové sítě, jejichž stavbu je možno navrhnout přímo v aplikaci, a jako formát hudební notace využívá MIDI. Ve fázi trénování, kdy je dostupná zvuková vlna i její ekvivalent v MIDI, se zvuk rozdělí na krátké úseky, pro něž je zjištěno, jaké noty v daném úseku hrají, a tyto dvojice jsou použity pro učení. Při testování sítě jsou stejným způsobem vybrány úseky testované zvukové vlny, které jsou následně zpracovány sítí. Výsledná data jsou následně spojena a uložena v MIDI.

3.1 Příprava zvukové stopy

Data určená jako vstup neuronové sítě by se měla blížit podobou tomu, jakým způsobem rozpoznává zvuk lidský sluch, neboť struktura neuronové sítě se blíží podobě lidského mozku a už kvůli samotné podstatě hudby jako lidského konceptu se vyplatí prezentovat vstup pomocí veličin, na něž je lidské rozpoznávání citlivé.

Z tohoto důvodu je výhodné reprezentovat zvuk nikoliv pomocí vzorků zvukové vlny, ale pomocí tzv. spektrogramu. Spektrogram je způsob vizualizace intenzity jednotlivých harmonických vln, z nichž je daný zvuk složen.



Obrázek 1: Ukázka zvukové vlny reprezentované křivkou a spektrogramem s lineární a logaritmickou škálou (Audacity).

Horizontální osa spektrogramu obvykle představuje čas, vertikální osa odpovídá frekvenčnímu prostoru zvuku (lze zobrazit na lineární nebo logaritmické škále, aby rozložení lépe odpovídalo lidskému vnímání hudby). Hodnota spektrogramu v daném čase a dané frekvenci je intenzita harmonické vlny o dané frekvenci nacházející se ve zvuku v daném čase.

Pro tvorbu spektrogramu lze využít Fourierovu transformaci pro převod funkce zvukové vlny do frekvenčního prostoru. Je potřeba brát v potaz fakt, že Fourierova transformace je aplikovatelná pouze na malý úsek celé zvukové vlny, tedy musí být opakována několikrát pro všechny úseky zvuku. Délka úseků musí být vybrána vhodně tak, aby zahrnovala dostatečné množství vzorků pro tvorbu celého

spektra, ale nesmí přesahovat dobu trvání jednotlivých uchem rozpoznatelných změn ve zvuku, nebo dojde k jeho „rozmazání“.

Pro převod škály spektrogramu z lineární na logaritmickou lze použít následující vzorec:

$$y = \frac{10^{\alpha x} - 1}{10^{\alpha} - 1}$$

Tento vzorec udává hodnotu v bodě y v nové škále pomocí bodu x v lineární škále (obojí normalizované na [0, 1]). Koeficient α ovlivňuje, jaká část frekvenčního spektra bude zabírat větší část nové škály (nižší frekvence pro vyšší hodnoty koeficientu a naopak). Pro $\alpha = 0$ lze vztah dodefinovat jako $y = x$.

Vstupem sítě nemůže být spektrogram celý (protože nemá pevnou délku), ale pouze jeho výřez o pevné délce. Jako vstup sítě může být použit jediný sloupec spektrogramu (tedy jedno spektrum) odpovídající přehrávaným notám nebo i jeho okolí (kontext). Přidání kontextu může zvýšit rozpoznávací schopnost sítě v situacích, kdy dochází k prudkým změnám ve zvukové vlně.

3.2 Reprezentace hudební notace

Nejdůležitějším parametrem hudby je soubor všech přehrávaných not, což je i výstup sítě použité v aplikace popsané touto prací. Pokud je podporován pouze monofonní zvuk (tedy zvuk, kde nehrají žádné tóny přes sebe), může se jevit jediná proměnná popisující aktivní notu jako postačující výstup sítě, ale přímé použití takové proměnné neodpovídá běžně používaným výstupním vrstvám. Trénování sítě se totiž snaží minimalizovat chybu v rozdílu odhadnutých hodnot a skutečných hodnot, a pokud by tyto hodnoty odpovídaly přímo číslům not, při výpočtech by hrálo číslo noty větší roli než pouze klasifikační – vyšší noty (s vyššími čísly) by měly jinou váhu než nižší noty.

Neuronové sítě se často využívají pro řešení klasifikačních problémů, kdy jednomu vstupu odpovídá jedna hodnota charakterizující daný vstup. Detekce noty v monofonním zvuku je také klasifikační problém, ale s velkým počtem kategorií. Pro reprezentaci těchto kategorií lze použít metodu zvanou one-hot encoding. V takovém případě má výstupní vektor sítě stejnou velikost jako počet kategorií a modelovaným výstupem sítě je vektor, kde jsou všechny hodnoty 0 až na tu, jíž index odpovídá dané kategorii (ta je 1).

Použití one-hot encoding vyžaduje jisté omezení hodnot výstupu. Pro tytu účely bývá nejčastěji použita vrstva aplikující na vstup funkci softmax. Ta převede vstup na soubor pravděpodobností odpovídající rozdělení náhodné veličiny výstupu sítě. Výstupní hodnoty jsou pouze v intervalu [0, 1] a jejich součet musí být 1. Výsledek této funkce lze interpretovat jako pravděpodobnost, tedy je-li element vektoru roven určité hodnotě, představuje tato hodnota pravděpodobnost, s jakou daná kategorie reprezentuje vstupní data.

I v takovém případě lze použít tuto síť na polyfonní zvuk, ale pak je nutné dodržet správnou podobu výstupu, aby mohl být hodnotou funkce softmax.

Zvláštní pozornost je potřeba věnovat reprezentaci ticha, tedy stavu, kdy není aktivní žádná nota. Jedna z možností je reprezentovat ticho jako samostatnou kategorii, ale v takovém případě (zvláště vzhledem k tomu, že ticho v MIDI neodpovídá tichu v samotném zvuku) může síť začít rozpoznávat ticho s větší pravděpodobností než jiné noty. Jednou možností je tak ticho vůbec nerozpoznávat (rozpoznaná hudební notace bývá většinou dodatečně ještě upravena a ticho tak může

být v relevantních částech odstraněno ručně) či generovat ticho v těch místech, kdy není žádná nota rozpoznána s dostatečně velkou pravděpodobností.

Alternativně lze pro výstupní vrstvu sítě použít (podobně jako u softmaxu) křížovou entropii, ale pouze mezi jednotlivými elementy výstupu a očekávaných dat (binární křížová entropie). V takovém případě musí být místo softmaxu použita jiná vrstva, jejíž výstup lze použít pro pravděpodobnosti, kupříkladu vrstva využívající funkci sigmoid. Tento přístup také nevyžaduje žádnou speciální reprezentaci ticha, neboť pak jej stačí reprezentovat nulovým výstupním vektorem.

3.3 Trénování a rozpoznávání

Nejdůležitější dva procesy pro fungování aplikace je trénování a rozpoznávání. Během trénování sítě je nutno vygenerovat spektrogram a spárovat jeho úseky s částmi vstupní hudby.

První komplikaci představuje fakt, že výstupem neuronové sítě má být stav hudby v daném časovém momentu, zatímco standard MIDI je založen na příkazech, tedy změnách stavu. Pro reprezentaci stavu hudby je dobré nahlížet na virtuální stroj schopný přijmu příkazů MIDI jako na jednoduchý procesor se sadou registrů pro každý kanál MIDI. Příkaz *Program Change* je interpretován jako změna jednoho registru obsahující číslo nástroj daného kanálu. Příkazy *Note On* a *Note Off* způsobí změnu jednoho ze 128 registrů určených pro noty daného kanálu.

Toto pojetí umožňuje velmi jednoduše procházet i tvořit soubory MIDI. Při procházení je každý příkaz interpretován jako změna registru a hodnoty všech registrů jsou v pevném intervalu předány řídícímu procesu. Naopak při tvorbě hudby může řídící kód nastavit každý takt registry podle výstupu sítě a zařídit vypsání pouze těch zpráv, které odpovídají změnám oproti předchozímu stavu všech registrů.

Zbývá nevyřešený problém výběru vhodné délky taktu, tedy doby trvání jednoho sloupce vstupního spektrogramu. Pro tyto účely bylo zvoleno rozlišení MIDI odpovídající výchozím parametrům, tedy $\frac{500000\ \mu m}{120} = 4{,}1\overline{6}\ \mathrm{ms}$. Tuto konstantu označíme jako dobu jednoho rámce (sloupce spektrogramu). Při standardní vzorkovací frekvenci 44100 Hz odpovídá 183,75 vzorkům. Aby se dala velikost jednoho rámce upevnit na 184 vzorcích a nedošlo k desynchronizaci mezi vstupním zvukem a MIDI, musela být vstupní frekvence zvuku navýšena na 44160 Hz, čehož bylo docíleno pouhým upravením parametrů nástroje ffmpeg.

Při trénování i testování je vyvolán ffmpeg na vstupním zvuku[8] a převede jej na posloupnost 16bitových vzorků s frekvencí 44160 Hz. Každých 184 vzorků tvoří jádro jednoho rámce. Aplikace umožňuje specifikovat navíc ještě velikost okraje rámce, což je oblast sdílená s ostatními rámci, která umožní zvýšení přesnosti a rozlišení spektrogramu. Udána v násobcích základní délky rámce, velikost okraje 2 způsobí, že pro Fourierovu transformaci bude použit nejen současný rámec, ale i dva předchozí a dva následující, tedy celková délka rámce bude 920 vzorků zaokrouhleno na nejbližší mocninu dvou (použitý algoritmus Fourierovy transformace pracuje nejlépe pro tyto velikosti), tedy na 1024 vzorků.

Než dojde k samotné transformaci do frekvenčního prostoru, je celý rámec vynásoben funkcí Hammingova okna. Ta má následující předpis (Harris, 1978):

[8] Vstupní může být aplikaci zadán explicitně jako zvukový soubor, případně může být vygenerován v situaci, kdy je k dispozici pouze soubor MIDI (pak je použit nástroj TiMidity++).

$$w(n) = a - (1 - a)\cos\frac{2\pi n}{N}; n = 0 \ldots N, a = 0{,}53836$$

Použití okna minimalizuje nepřesnosti způsobené okraji rámce a zvýrazní jeho střed (Oppenheim, 1999).

Takto upravený rámec je dále zpracován inverzní Rychlou Fourierovou transformací (iFFT) a použit jako jeden sloupec spektrogramu. Všechny hodnoty ve spektrogramu jsou navíc ještě logaritmicky upraveny, aby lépe reprezentovaly lidské vnímání hlasitosti, pomocí tohoto vztahu:

$$v_o = \log_{10}\frac{v_i}{v_s}$$

Nastavitelný parametr v_s představuje práh slyšitelnosti, tedy intenzitu, při níž výstupní hodnota v_o bude 0. Intenzity v_i pod prahem slyšitelnosti nejsou do spektrogramu zahrnuty.

Rámec se svým kontextem (okolními sloupci), jehož velikost je nastavitelná při tvorbě sítě, představuje celý vstup pro neuronovou síť v jeden časový moment.

Spektrogram vstupního zvuku je při trénování sítě spárován s hudebním stavem vzorového souboru MIDI. Jak již bylo uvedeno výše, příkazy obsažené v souboru MIDI jsou interpretovány jako změny interního stavu virtuálního zařízení MIDI a snímky tohoto stavu jsou každých $4{,}1\overline{6}$ ms použity jako očekávaný výstup sítě. Ačkoliv použité rozlišení nepředstavuje pevný limit a teoreticky může být použit i soubor obsahující jemnější rozlišení, takto malé hodnoty nepředstavují pro lidské vnímání rozdíl. Pokud vstupní soubor přesně odpovídá očekávaným hodnotám časování, technicky jsou hudební stavy posunuty oproti zvukovým rámcům o polovinu rozlišení (ke změnám stavů dochází právě na okrajích rámců, zatímco spektrum odpovídající rámci reprezentuje celý interval), ale ani tato malá výchylka není v porovnání s běžnou délkou hudby nijak problematická.

Jako trénovací výstup neuronové sítě není použit přímo stav zařízení MIDI, ale jeho reprezentace zvolená podle výstupní metody sítě. Pokud je zvolena výstupní metoda softmax, je napřed zjištěn počet aktivních not c a následně je všem těmto notám přiřazena pravděpodobnost $\frac{1}{c}$. Není-li aktivní žádná nota, všechny pravděpodobnosti jsou nastaveny na $\frac{1}{128}$. Při použití metody sigmoid jsou nastaveny pravděpodobnosti aktivních not na 1 a ostatní na 0.

Rozpoznávání libovolného zvuku připraví zvuková data pro neuronovou síť zcela stejným způsobem. Po průchodu dat sítí je výstup analyzován a použit k nastavení odpovídajícího stavu virtuálních registrů MIDI. Jelikož ve výstupu je obsažena i pravděpodobnost, s jakou byla daná nota rozpoznána, je tato pravděpodobnost použita k určení, zdali bude nota akceptována jako výstupní. Toto porovnání je dvojí: absolutní (pravděpodobnost je porovnána s pevnou hodnotou) a relativní (pravděpodobnost je porovnána s násobkem největší rozpoznané pravděpodobnosti). Tímto způsobem jsou odmítnuty jak obecně příliš nepravděpodobné noty, tak i noty, které jsou přebity pravděpodobnějšími notami.

3.4 Testování přesnosti

Pro účely testování přesnosti sítě podporuje aplikace možnost testování na náhodně vytvořené hudbě či na existujícím souboru MIDI. V takovém případě je

potřeba stanovit, jakým způsobem bude porovnáván výstup sítě se skutečným hudebním stavem.

Jsou-li k dispozici dva hudební stavy (rozpoznaný a původní), nabízí se možnost počítat přesnost jako podíl situací, kdy byla správně rozpoznána přítomnost noty, ku všem těmto situacím (těch je 128násobek délky hudby v základních taktech). Jelikož je ale zpravidla výstupem jediný nejpravděpodobnější tón, ve většině případů bude korektně rozpoznáno ticho a takto počítaná přesnost (především při použitý metody softmax) by dosahovala 99 %.

Z tohoto důvodu je při počítání přesnosti v aplikaci ignorována situace, kdy je úspěšně rozpoznáno ticho, a do výsledného podílu jsou započítány pouze situace, kdy se úspěšně rozpoznala aktivní nota, kdy se nerozpoznala aktivní nota a kdy se rozpoznala neexistující nota, jinak řečeno celková přesnost je podílem mohutnosti průniku původních a rozpoznaných not ku mohutnosti jejich sjednocení. Případy neúspěchu jsou si rovnocenné, tedy nenalezená nota představuje stejně velkou chybu jako nalezená neexistující nota. Tento podíl je však aplikován pouze na jednotlivé základní intervaly hudební stopy, neboť jinak by celkovou přesnost ovlivňovaly více ty úseky hudby, kde hraje mnoho not zároveň, nebo naopak ty úseky, kde je ticho a síť kvůli tomu rozpozná not více, přestože pro posluchače jsou tyto úseky stejně důležité jako jiné části hudby stejné délky. Z toho důvodu je celková přesnost rozpoznání počítána jako aritmetický průměr výše zmíněných podílů pro každý základní interval.

3.5 Automatické generování vstupu

Narozdíl od rozpoznávání řeči, kdy databáze vzorků jsou omezené, představuje hudba typ dat, která se dají velmi jednoduše generovat. Především, jsou-li k dispozici vzorky hudby ve formátu MIDI, mohou být tyto vzorky převedeny mnoha způsoby do skutečné zvukové vlny. Jak již bylo uvedeno v kapitole o standardu MIDI, pro reprodukci zvuku je potřeba soundfont, přiřazující všem nástrojům jejich charakteristický zvuk. Díky tomu může jedno hudební dílo znít různě na různých zařízeních, pokud používají jiný soundfont, a díky tomu lze také připravit několik verzí testovacích dat, kde každá bude obsahovat jinou zvukovou stopu piana.

Druhá výhoda hudby a zvoleného formátu je možnost automatické tvorby vstupu pro síť a tedy odstranění nutnosti hledat vzorky hudby. Je-li zařízena dostatečná variabilita vytvořené hudby, je vstup sítě srovnatelný s lidmi vytvořenou hudbou. Do takto generované hudby lze i po převedení do zvukové vlny vnést další změny jako dodatečný šum, ambientní zvuky, řeč a jiné efekty, které mohou při rozpoznávání snížit přesnost sítě, ale pokud jsou použity při trénování, mohou zvýšit její adaptabilitu.

3.6 Rozlišování nástrojů

Doposud byly popsány procesy, kde použité nástroje nehrají roli, ale všechny se dají použít i v případě, chceme-li rozpoznávat různé nástroje a ne pouze jeden. V takovém případě je pro každý nástroj vytvořena separátní síť, přičemž všechny tyto sítě jsou vytvořeny se stejnou strukturou. V celém systému sítí poté může existovat síť pouze pro piano, síť pouze pro flétnu, síť pouze pro housle atd.

Při trénování jsou vybrány ty sítě, které odpovídají aktuálně hrajícím nástrojům, a každé z nich jsou poskytnuty informace pouze o nástroji, který rozpoznává. Ostatní hrající nástroje v takovém případě představují pouze šum.

Při rozpoznávání jsou vstupní data postupně předána všem sítím a jejich výstup je spojen. V takovém případě je pro každý rozpoznaný nástroj využit volný kanál MIDI a všem použitým kanálům je nastaven nástroj, jehož síť rozpoznala noty z těchto kanálů.

3.7 Polyfonie

Podobným problémem jako několik nástrojů hrajících přes sebe je jeden nástroj hrající více tónů zároveň. V takovém případě jsou při trénování sítě všechny tóny rovnocenně započteny, ale není zaručeno, že je dokáže síť úspěšně zkombinovat při rozpoznávání.

Výstupní metoda softmax představuje určitý problém při rozpoznávání, neboť je primárně určena pro jedinečnou klasifikaci, ale pokud je výstupem sítě několik hodnot s vyšší pravděpodobností, mohou být všechny použity jako výstup aplikace.

3.8 Vylepšení přesnosti sítě

Častým problémem vzniklým při trénování sítě je tzv. overfitting (přeučení). K němu dochází v situaci, kdy je síť naučena rozpoznávání příliš konkrétní podoby dat a nedokáže z ní vyvodit obecné charakteristiky. V takovém případě síť není schopna přesně rozpoznávat data sdílející stejné klíčové charakteristiky, ale mající trochu jinou podobu.

Je mnoho způsobů, jak overfittingu předejít. Některé metody učení sítě přidávají do chybové funkce člen, který zabraňuje příliš nízkým chybám. Obecně je výhodné zařídit, aby síť dostávala dostatečně rozsáhlá a rozmanitá data a nebylo jí umožněno se příliš zaměřit na jednu konkrétní podobu dat.

Pro tyto účely umožňuje aplikace trénovat síť na náhodném uspořádání původních vstupních dat. V takovém případě se nemůže síť zcela přizpůsobit jednomu tónu, hraje-li příliš dlouho.

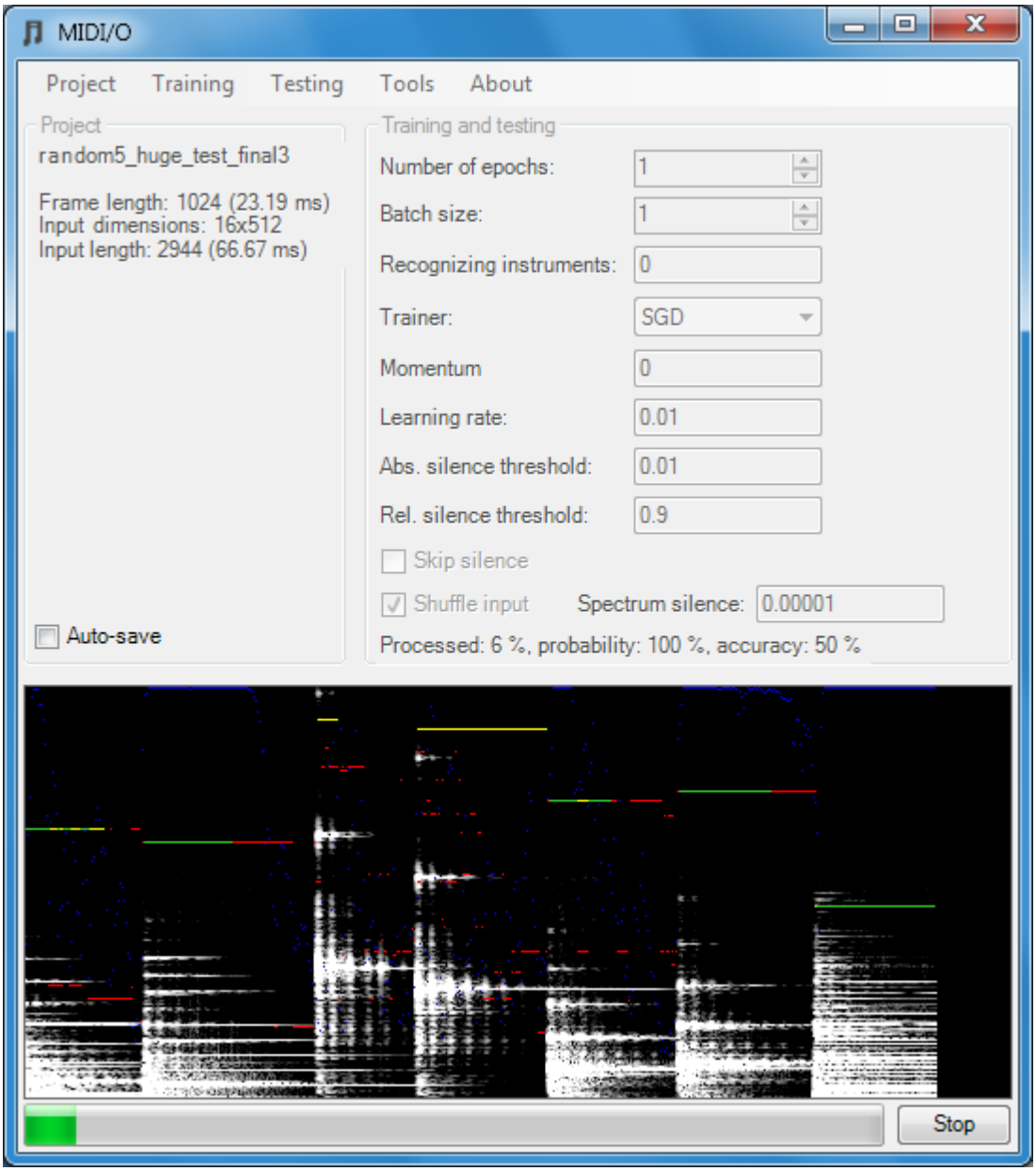
4. Ovládání a vývoj aplikace

4.1 Uživatelské prostředí

Před použitím aplikace je nutno upravit konfigurační soubor XML aplikace (MIDIO.xml) a nastavit správné cesty k nástrojům ffmpeg a TiMidity++, pokud nejsou součástí instalace. Výchozí konfigurační soubor vypadá takto:

```
<?xml version="1.0" encoding="UTF-8"?>
<MIDIO>
  <Training>
    <Trainer>SGD</Trainer>
    <NumEpochs>1</NumEpochs>
    <LearningRate>0.01</LearningRate>
    <Momentum>0</Momentum>
    <BatchSize>1</BatchSize>
  </Training>
  <Input>
    <Shuffle>true</Shuffle>
    <SkipSilence>false</SkipSilence>
    <SpectrumSilenceMinimum>0.00001</SpectrumSilenceMinimum>
  </Input>
  <Output>
    <RelativeSilenceThreshold>0.9</RelativeSilenceThreshold>
    <AbsoluteSilenceThreshold>0.01</AbsoluteSilenceThreshold>
  </Output>
  <Ffmpeg>
    <Location>tools\ffmpeg.exe</Location>
  </Ffmpeg>
  <TiMidity>
    <Location>tools\timidity.exe</Location>
  </TiMidity>
  <Google>
    <Credentials></Credentials>
  </Google>
</MIDIO>
```

Po spuštění aplikace se zobrazí hlavní okno pro rozpoznávání:



Obrázek 2: Ukázka okna aplikace v testovacím režimu.

Toto okno je rozděleno do tří hlavních částí. Oblast „Project“, vyplněná po načtení či vytvoření projektu, obsahuje základní informace o síti a rozpoznávání (které již po vytvoření projektu nelze změnit). Hodnota „Frame length“ je délka rámce spektrogramu ve vzorcích a milisekundách, „Input dimensions“ jsou rozměry vstupu neuronové sítě (části spektrogramu) a „Input length“ je délka tohoto úseku spektrogramu ve vzorcích a milisekundách.

Oblast „Training and testing“ umožňuje nastavení všech parametrů trénování a rozpoznávání sítě. Prvek „Number of epochs“ nastavuje, kolikrát při trénování aplikace projde trénovacími daty. „Batch size“ představuje velikost dávky, která bude předána neuronové síti při trénování jako celek. Pole „Recognizing instruments“ umožňuje nastavit, jaké nástroje budou rozpoznávány sítí, a dojde-li k jeho změně, vytvoří se nové sítě a nepoužívané se odstraní. „Trainer“ ovlivňuje, jaký trénovací algoritmus bude použit (SGD nebo Adam). Pole „Momentum“ a „Learning rate“ nastavují parametry trénování použité k minimalizaci chyby (pro algoritmus Adam jsou jejich názvy „Momentum decay“ a „Learning rate“). Pole

„Abs. silence threshold“ a „Rel. silence threshold“ specifikují, jaké rozpoznané noty budou použity při tvorbě výstupu. Pravděpodobnost rozpoznané noty musí být vyšší než absolutní hranice a zároveň vyšší než násobek relativní hranice a nejvyšší rozpoznané pravděpodobnosti.

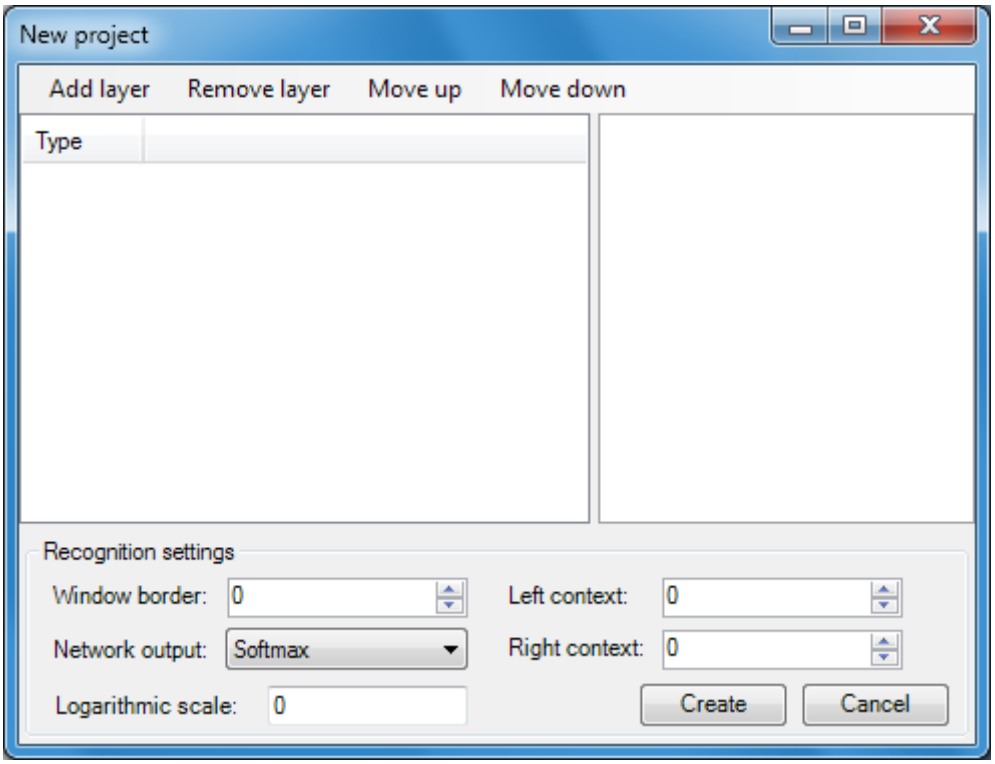
Přepínač „Skip silence“ zařídí přeskočení tichých míst v hudbě při trénování sítě. Přepínač „Shuffle input“ způsobí při trénování náhodné rozmístění trénovacích dat. Pole „Spectrum silence“ specifikuje práh slyšitelnosti zvuku při počítání logaritmické intenzity.

Hlavní menu programu obsahuje v sekci „Project“ nástroje pro vytvoření, načtení či uložení projektu. Sekce „Training“ obsahuje nástroje pro učení neuronové sítě na různých vstupech, tedy na jednom vstupním souboru MIDI, na vstupním souboru MIDI a jeho odpovídající zvukové reprezentaci a na náhodně vygenerované hudbě. Sekce „Testing“ umožňuje aplikování neuronové sítě na libovolná vstupní data za účelem vytvoření souboru MIDI a také obsahuje nástroje pro testování přesnosti sítě na existujícím souboru MIDI či na náhodně vygenerovaném. V sekci „Tools“ je možnost spojení většího množství vzorků hudby do jedné stopy, což lze použít zároveň s náhodně rozmístěnými testovacími daty, a také nástroj na rozpoznávání hlasu (využívá Google Speech-to-Text) a sekce „About“ zobrazí informace o programu.

V dolní polovině okna se nachází graf popisující právě probíhající trénování či testování sítě. Bíle či šedě zbarvené pixely korespondují s hlasitostí daného místa spektrogramu (ticho je znázorněno černě), přičemž pixely umístěny dole popisují nižší frekvence než pixely nahoře. Při trénování či tvorbě MIDI jsou použity pouze zelené pixely, které označují noty (původní či rozpoznané). Výše umístěný pixel představuje vyšší notu. Pokud je síť nastavena na rozpoznávání, objevují se také modré pixely představující pravděpodobnost správného rozpoznání noty v daném momentě (vyšší pixel odpovídá vyšší pravděpodobnosti).

Pokud je testována přesnost sítě (na zadané či náhodné hudbě), kromě zelených pixelů se zobrazují také pixely žluté a červené. Zelené pixely v tomto případě představují noty, které se nacházejí jak v původní hudbě, tak i ve výstupu sítě. Červené pixely odpovídají notám, které byly rozpoznány, ale chybně (nejsou v původní hudbě). Žluté pixely představují noty, které byly v původní hudbě, ale síť je nerozpoznala.

Vytvoření nového projektu probíhá v separátním okně:



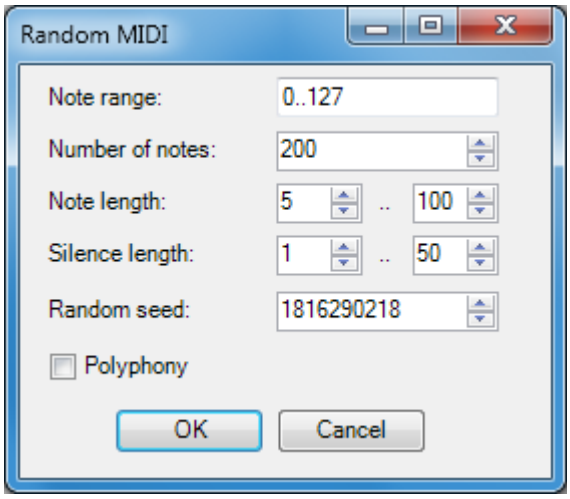
Obrázek 3: Vytváření nového projektu.

V tomto okně lze navrhnout neuronovou síť a všechny parametry nutné pro určení podoby vstupu a výstupu. Pod položkou „Add layer“ se nachází menu pro přidání různých typů vrstev a jejich parametrů. Nově přidané vrstvy se zobrazují v levém panelu. Vrstvy se dají odstranit tlačítkem „Remove layer“ či posouvat tlačítky „Move up“ a „Move down“. Je-li vybrána vrstva, v panelu napravo se zobrazí její vlastnosti.

Pole „Window border“ určuje velikost okraje rámce spektrogramu (v násobcích základní délky), pole „Left context“ a „Right context“ velikost kontextu vlevo a vpravo od rámce. Volba „Network output“ má možnosti „Softmax“ a „Sigmoid“ určující, jakým způsobem bude počítán a interpretován výstup sítě. „Logarithmic scale“ určuje koeficient použitý při transformaci vstupu na logaritmickou škálu.

Jsou-li všechny parametry správné, tlačítkem „Create“ je vytvořen nový projekt. Pokud je aplikaci zadán projekt (nový či načtený), je možné provádět se sítí další operace jako trénování na vstupní hudbě či generování souboru MIDI ze zvukového záznamu.

Aplikace je schopna pomocí tlačítek v hlavním menu vytvořit náhodnou hudbu postačující pro rychlé učení sítě. Parametry hudby se nastavují pomocí dialogového okna:

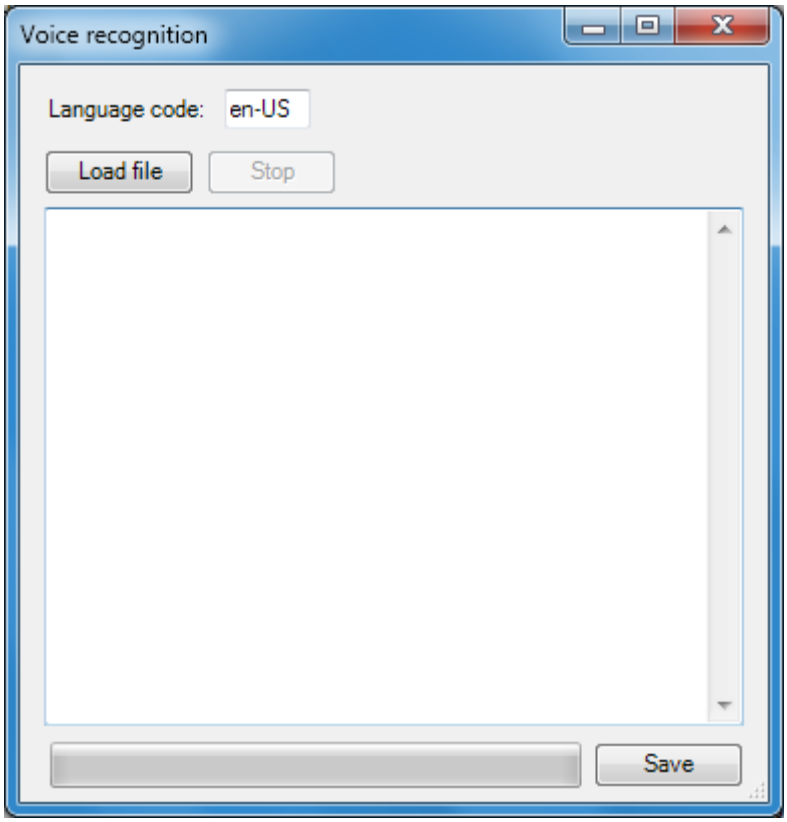


Obrázek 4: Generování náhodných dat.

„Note range“ představuje rozsah not obsažených v generované hudbě. Lze v něm specifikovat rozsah pomocí dvou okrajových not jako *a*..*b* či jako seznam not oddělený čárkami či mezerami. Náhodná hudba obsahuje základní počet not daný polem „Number of notes“. Délka každé noty je náhodně vybrána v intervalu „Note length“ a každou notu předchází ticho v náhodné délce „Silence length“. Parametr „Random seed“ specifikuje seed pro inicializaci generátoru pseudonáhodných čísel použitý pro generování hudby. Volba „Polyphony“ náhodně přes existující noty vygeneruje další noty navíc.

Při trénování sítě se v oblasti „Training and testing“ zobrazuje průměrná chyba sítě při trénovacích datech. Při testování přesnosti sítě se ukazuje parametr přesnosti popsaný výše. Pokud je síť aplikována na zvuková data (testování či rozpoznávání), je zobrazena také informace o pravděpodobnosti výsledku.

Vedlejší nástroj obsažený v aplikaci je rozpoznávání hlasu pomocí Google Speech-to-Text. Aby mohl být nástroj použit, je potřeba v konfiguraci MIDIO.xml vyplnit element Google/Credentials platnou cestou k souboru .json obsahujícímu klíče pro přístup do Google Cloud. Pokud je soubor zadán, rozpoznávání hlasu probíhá ve zvláštním okně:



Obrázek 5: Rozpoznávání hlasu.

Pro úspěšné rozpoznání je nutno zadat do pole „Language code“ kód jazyka podle standardu ISO 639-1. Po načtení souboru je zkontaktována služba Speech-to-Text a v hlavním textovém poli se začne objevovat přepis řeči. Ten je možno uložit jako titulky ve formátu SubRip (.srt).

4.2 Výběr jazyka a technologií

Výběr jazyka je do velké míry podmíněn knihovnami, které jsou v daném jazyce dostupné pro řešení daného problému, ale přestože v oblasti neuronových sítí a hlubokého učení je většinou používán jazyk Python, byl pro tuto aplikaci zvolen jazyk C# a platforma .NET Framework. Důvody jsou následující:

- Osobní důvod – s jazykem C# mám mnohem větší zkušenosti než s Pythonem, který neovládám.
- C# je kompilovaný a hotové aplikace lze lehce distribuovat, oproti Pythonu, kde pro využití podobných aplikací bývá většinou potřeba mít inicializované prostředí Pythonu a jeho interpretr. .NET Framework je dostupný na všech posledních operačních systémech Windows a na jiných platformách lze použít prostředí Mono pro spuštění aplikací v něm.
- Výsledná aplikace má být běžná okenní aplikace, což prostředí .NET Framework nabízí bez nutnosti instalace externích knihoven.

Primární požadavky aplikace na technologie jsou dva: podpora neuronových sítí, matematická knihovna pro transformaci vstupních dat a podpora práce se soubory ve formátu MIDI.

Populární volbou pro tvorbu neuronových sítí je open-source knihovna TensorFlow[9] od týmu Google Brain, navržená pro využití v jazyce Python. Přestože existuje port této knihovny pro C# (TensorFlowSharp[10]), tento port v době tvorby aplikace (2018/2019) podporoval pouze aplikování neuronových sítí s předem vytvořeným modelem, což je v rozporu s tématem aplikace, jejíž nedílnou součástí musí být i trénování neuronové sítě.

Microsoft poskytuje dvě velké knihovny pro práci s neuronovými sítěmi – CNTK[11] a ML.NET[12]. CNTK (Cognitive Toolkit) podporuje trénování neuronových sítí včetně velkého množství parametrů, ale je též primárně navržena pro Python a dokumentace pro C# není kompletní, takže jsem preferoval nalézt přístupnější řešení. ML.NET je relativně nový projekt, který ale v současné fázi podporuje pouze omezenou škálu aplikací, primárně pouze klasifikační úlohy na obrázcích.

Nakonec byla pro aplikaci zvolena knihovna ConvNetSharp[13] (vycházející z ConvNetJS pro JavaScript), která umožňuje velmi lehce a přímočaře vytvořit a testovat konvoluční neuronovou síť. Tato knihovna navíc také podporuje běh na grafickém procesoru pomocí CUDA (Compute Unified Device Architecture).

Pro aplikaci Fourierovy transformace a Hammingova okna, použitých při převodu zvukové vlny na spektrogram, je využita knihovna MathNet[14].

Na čtení a tvorbu souborů MIDI existují různé knihovny pro .NET, ale mnoho z nich neimplementuje všechny požadované prvky, jako např. podporu více stop, rozlišování kanálů, správnou interpretaci tempa apod. Součástí aplikace je vlastní implementace základních prvků formátu SMF, které jsou potřeba, podle specifikace.

Aplikace využívá externí program TiMidity++[15] pro tvorbu zvukové vlny ze souboru MIDI a nástroj ffmpeg[16] pro čtení zvuku z libovolných zvukových souborů.

4.3 Architektura aplikace

Aplikace (zdrojové kódy jsou součástí přílohy A.1 Zdrojové kódy aplikace) je rozdělena do několika jmenných prostorů podle jejich primárních funkcí. Jmenný prostor *App* zahrnuje všechna okna a dialogy poskytované aplikací. Jmenný prostor *Midi* obsahuje funkce pro čtení, zápis a interpretaci souborů MIDI. Ve jmenném prostoru *Wave* se nacházejí třídy určené pro čtení zvukových dat a transformací zvukových vln. Jmenný prostor *Recognizer* obsahuje všechny funkce pracující s neuronovou sítí, tedy trénování i rozpoznávání.

Kořenový jmenný prostor aplikace obsahuje třídu *Program* se spouštěcím kódem programu, třídu *Settings* pro čtení a ukládání nastavení aplikace ve formátu

---

[9] https://www.tensorflow.org/

[10] https://github.com/migueldeicaza/TensorFlowSharp

[11] https://www.microsoft.com/en-us/research/product/cognitive-toolkit/

[12] https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet

[13] https://github.com/cbovar/ConvNetSharp

[14] https://www.mathdotnet.com/

[15] http://timidity.sourceforge.net/

[16] https://ffmpeg.org/

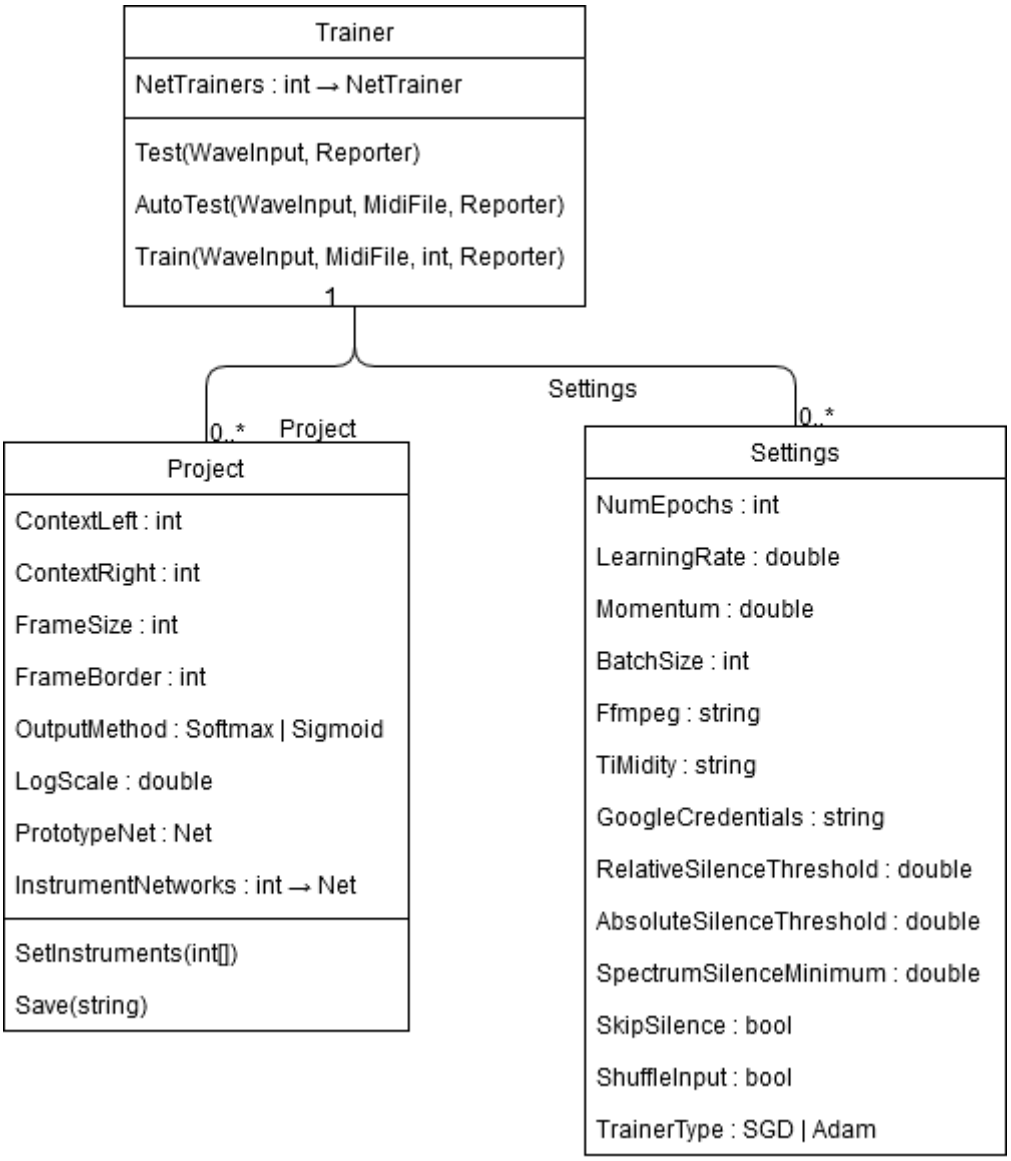
XML (které je použito i v dalších třídách) a třídu *Constants* obsahující pevně dané konstanty jako použitou frekvenci zvuku či rozlišení MIDI.

Projekt, který aplikace umožňuje vytvořit a načítat, je reprezentován třídou *Project*. Ta obsahuje prototypovou síť, která se duplikuje pro každý nový rozpoznávaný nástroj, a různé další parametry, které se nastavují při vytváření projektu a jsou dále neměnné.

Jádro aplikace představuje třída *Trainer*, která uchovává kolekci objektů z knihovny ConvNetSharp pro trénování všech sítí z třídy *Project*, které se vytvoří vždy na začátku trénování podle zadané metody (SGD nebo Adam) a parametrů.

Pro čtení zvukové vlny existuje ve jmenném prostoru *Wave* třída *WaveInput*. Ta může být inicializována jak pro čtení z běžného zvukového souboru, tak pro reprodukci zvuku ze souboru MIDI. Metoda *Frames* vrací posloupnost zvukových rámců ze vstupu jako sloupce spektrogramu. Metoda *AddContext* slouží k výběru rámců z posloupnosti společně s jejich kontextem, tedy specifikovatelným množstvím předchozích a následujících rámců.

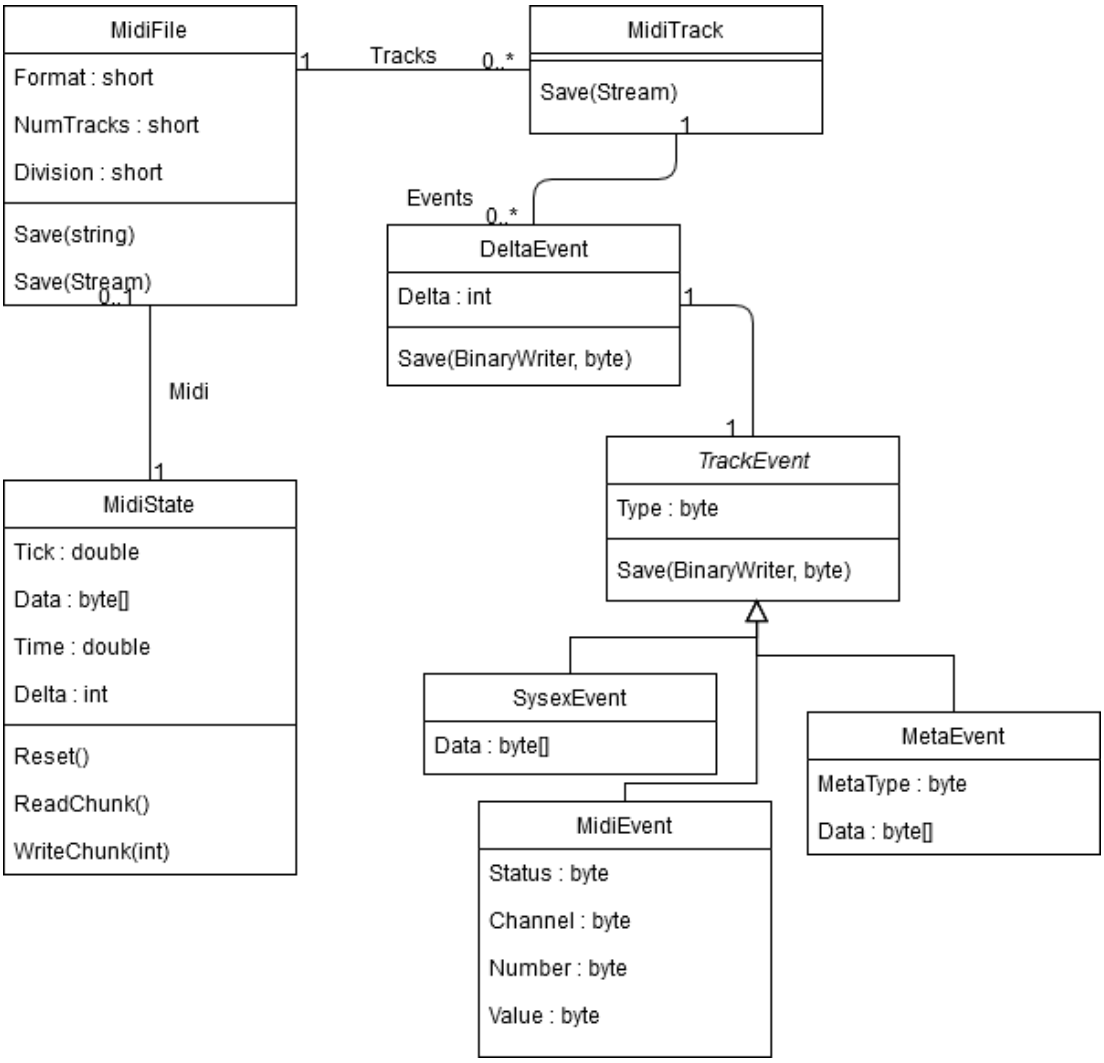
Třída *WaveInput* vytváří třídu *WaveTransformation* pro aplikování Fourierovy transformace na posloupnost vzorků.



Obrázek 6: Klíčové komponenty aplikace.

Základní třídou pro práci se soubory MIDI je třída *MidiFile*. Ta podporuje načtení ze souboru či inicializaci nového souboru (pak je parametrem jeho časování). Po otevření souboru jsou postupně procházeny všechny uložené stopy a přidávány do seznamu stop. Jedna stopa, tedy posloupnost příkazů, je reprezentovaná třídou *MidiTrack*. Stuktura *DeltaEvent* reprezentuje příkazy stopy, obsahuje čas od předchozího příkazu i samotný typ příkazu určený abstraktní třídou *TrackEvent*. Tuto třídu dědí další třídy ve jmenném prostoru *Midi.Events*, konkrétně *MidiEvent* (příkaz nastavující parametry hudby), *MetaEvent* (příkaz ovlivňující přehrávání či přehrávač) a *SysexEvent* (speciální příkazy pro různé systémy a výrobce).

Třída *MidiState* představuje implementaci virtuálního procesoru MIDI pro účely převodu příkazů na hudební stav. Všechny registry jsou obsaženy v poli bajtů *Data*, jež je modifikováno zpracovanými příkazy. Toto pole obsahuje úseky pro všech 16 kanálů a každý kanál má registry pro nástroj, hlasitost a stav všech 128 not. Pro tvorbu zpráv je uchovávána kopie všech registrů a po všech modifikacích na konci každého časového úseku je řídící kód informován o nové sadě registrů.



Obrázek 7: Komponenty reprezentující architekturu MIDI.

5. Evaluace

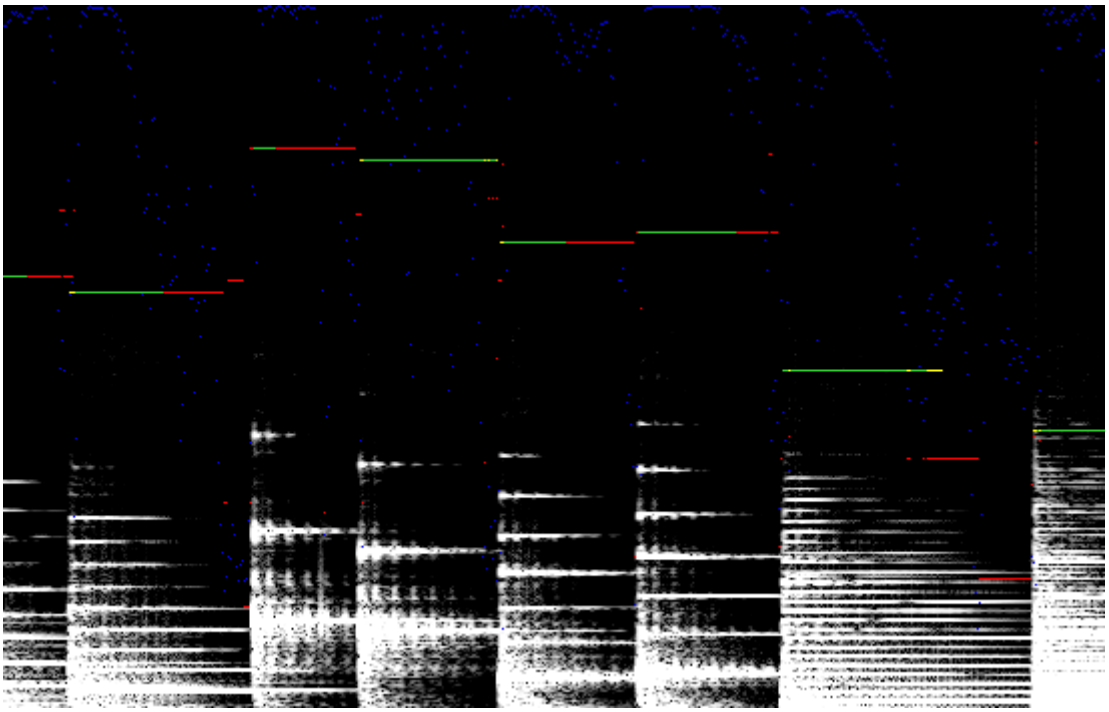
Pro testování základních funkcí sítě byl použit model s následujícími parametry:

Window border	2
Left context	15
Right context	0
Network output	Softmax
Logarithmic scale	0
Batch size	1
Trainer	SGD
Momentum	0.5
Learning rate	0.01
Abs. silence threshold	0.01
Rel. silence threshold	0.9
Spectrum silence	0.00001
Shuffle input	

Skryté vrstvy navržené neuronové sítě:

- `ConvLayer(Width=5, Height=5, FilterCount=3, Stride=1)`
- `TanhLayer`
- `DropoutLayer(DropProbability=0.5)`
- `PoolLayer(Width=2, Height=2)`
- `ConvLayer(Width=3, Height=3, FilterCount=1, Stride=1)`
- `TanhLayer`
- `DropoutLayer(DropProbability=0.5)`
- `PoolLayer(Width=2, Height=2)`

Architektura této sítě se shoduje s modelem, který použili Verma a kol. ve své práci a který pochází z modelů navržených pro klasifikaci datasetu MNIST. Tato síť byla trénována na souboru 1000 náhodně vygenerovaných not v rozsahu 21 až 108 s výchozími dalšími parametry generování a náhodným generátorem inicializovaným hodnotou 363351996. Pro účely trénování bylo vypnuto trénování na tichu a povoleno náhodné pořadí vstupních dat.

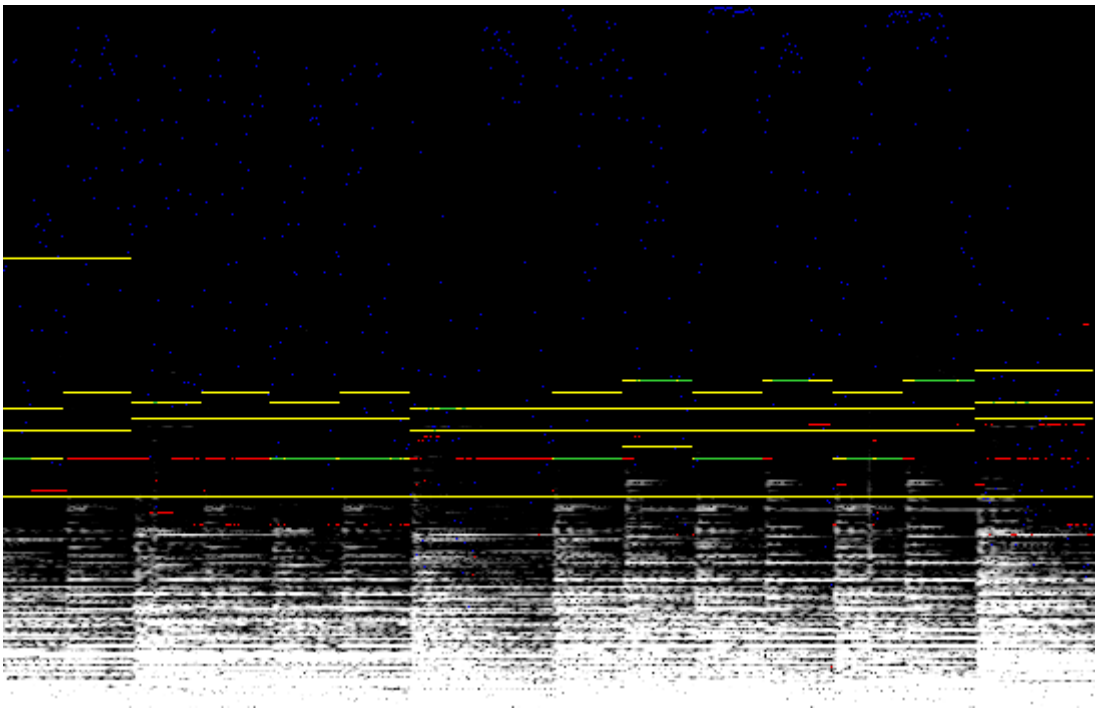


Obrázek 8: Ukázka výkonu první sítě na náhodně generovaných datech se stejným soundfontem.

Při testování na náhodném vzorku monofonních dat dochází k průměrné přesnosti 60 %.

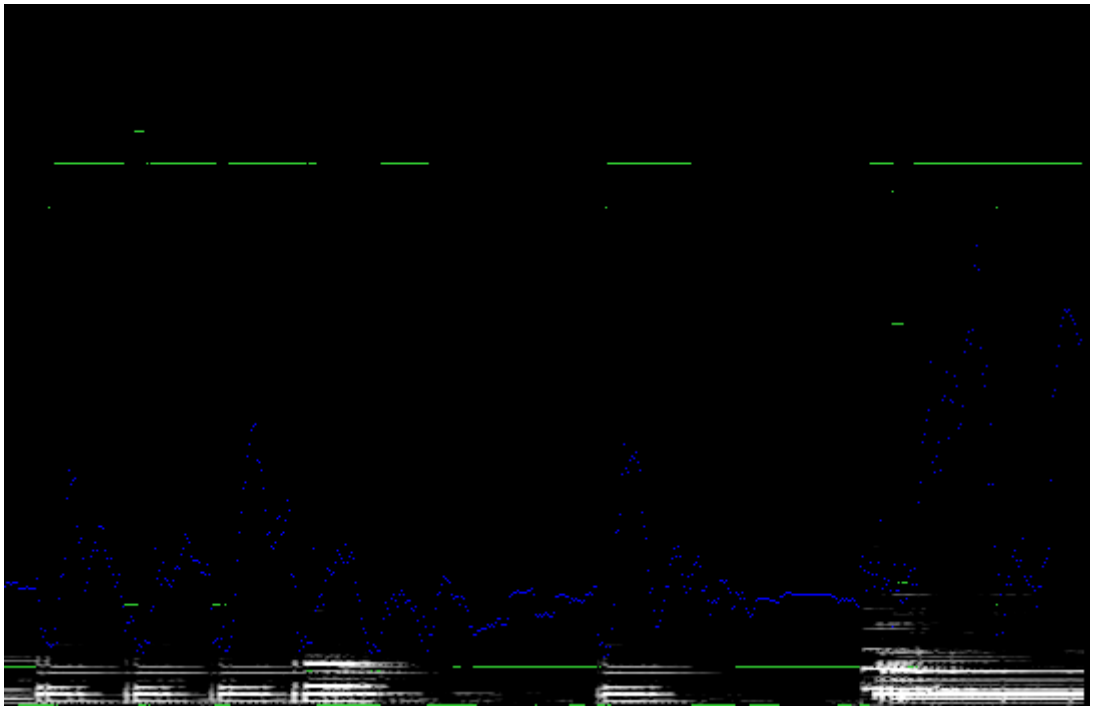
Nejčastějším případem chyby sítě je doběh tónu, tedy úsek zvuku, kde v hudebním záznamu již žádná nota aktivní není, ale nástroj je stále slyšet. Jak je patrné ze spektrogramu, mezi jednotlivými úseky před koncem noty a po ní není velký rozdíl, a síť tedy nemá dobré indicie, aby mohla určit, kde nota končí.

Při testování sítě na polyfonní hudbě (ale stále reprodukované strojově se stejným soundfontem) nedokáže tato síť spolehlivě rozpoznat všechny hrající tóny, ale zpravidla rozezná alespoň jeden z nich.



Obrázek 9: Ukázka výkonu první sítě na polyfonní hudbě se stejným soundfontem (MAPS_MUS-alb_se3_AkPnBcht.mid).

Při testování sítě na hudbě s jiným soundfontem či na reálných nahrávkách (ovlivněných šumem či kompresí) tato síť, podle očekávání, není schopna správně rozeznat jednotlivé tóny, neboť byla trénována na konkrétní podobě piana.



Obrázek 10: Ukázka výkonu první sítě na reálné hudbě (pop.ogg).

Pro účely testování výkonu aplikace na rozličnější hudbě byl zvolen dataset MAPS (MIDI Aligned Piano Sounds)[17], obsahující ukázky piana společně s jejich přepisy do MIDI. Z ní byly vybrány pouze zvuky z kategorie ISOL/CH, tedy posloupnosti krátkých tónů se zvyšující se výškou, a to z důvodu, aby nedošlo k prioritizování některých tónů. Všechny ukázky byly následně spojeny pomocí nástroje v aplikaci do jedné stopy.

Byly použity tyto parametry aplikace:

Window border	1
Left context	8
Right context	7
Network output	Softmax
Logarithmic scale	5
Batch size	64
Trainer	Adam
Momentum decay	0.00001
Learning rate	0.001
Abs. silence threshold	0.01
Rel. silence threshold	0.9
Spectrum silence	0.0005
Shuffle input	

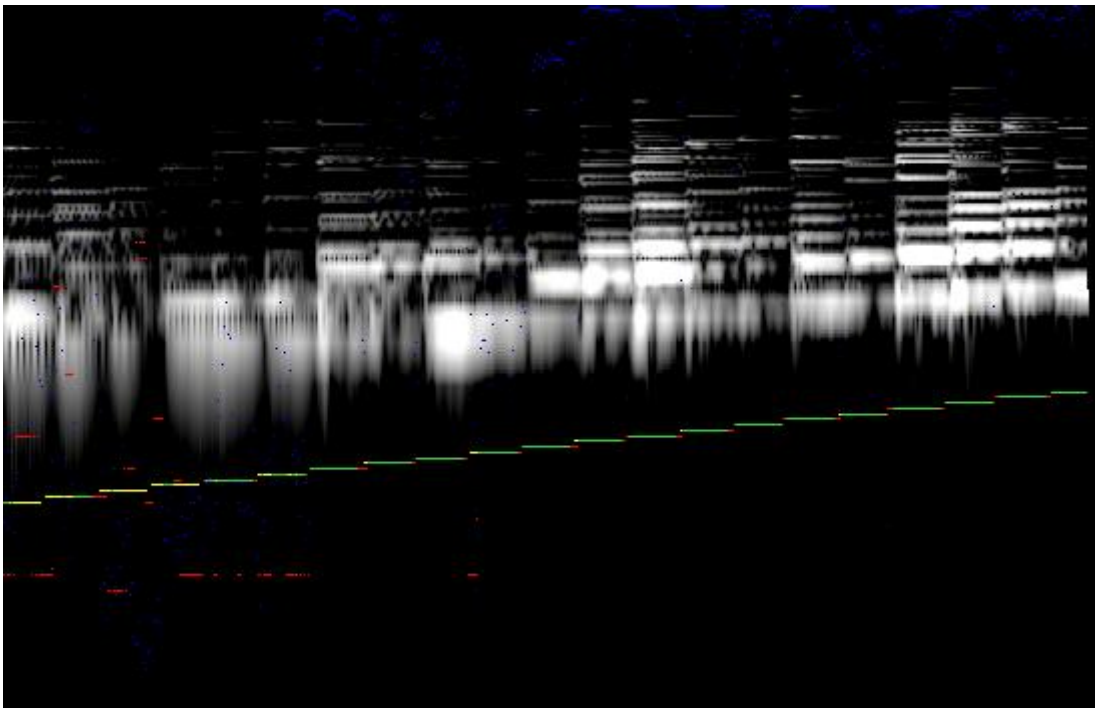
Skryté vrstvy navržené neuronové sítě:

- `ConvLayer(Width=5, Height=5, FilterCount=1, Stride=1)`
- `TanhLayer`
- `PoolLayer(Width=2, Height=2)`
- `ConvLayer(Width=3, Height=3, FilterCount=1, Stride=1)`
- `TanhLayer`
- `PoolLayer(Width=2, Height=2)`
- `FullyConnLayer(NeuronCount=256)`

Dále bylo při trénování zapnuto trénování na tichu a povoleno náhodné pořadí vstupních dat.

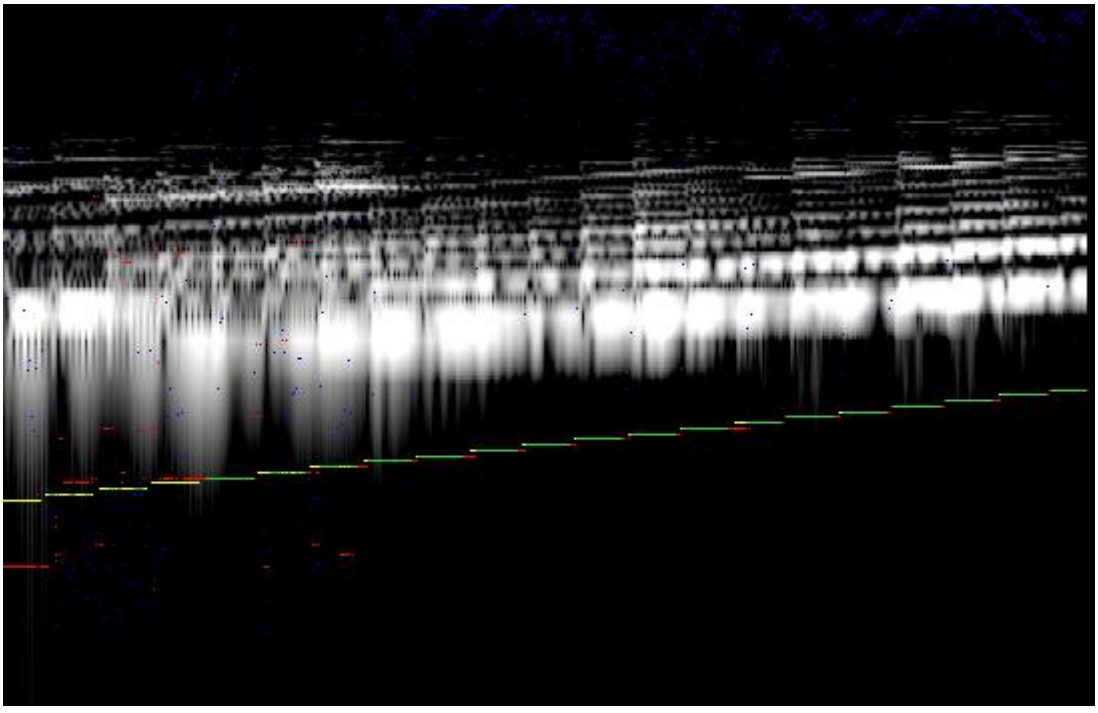
Po 30 epochách trénování dosahuje tato síť průměrné přesnosti 71 % při testování na části původních trénovacích dat (MAPS_ISOL_CH0.1_F_AkPnBcht), přičemž k největšímu poklesu úspěšnosti dochází při nejnižších frekvencích, kde není spektrogram natolik výrazný:

[17] https://www.tsi.telecom-paristech.fr/aao/en/2010/07/08/maps-database-a-piano-database-for-multipitch-estimation-and-automatic-transcription-of-music/



Obrázek 11: Ukázka výkonu druhé sítě na vzorové hudbě.

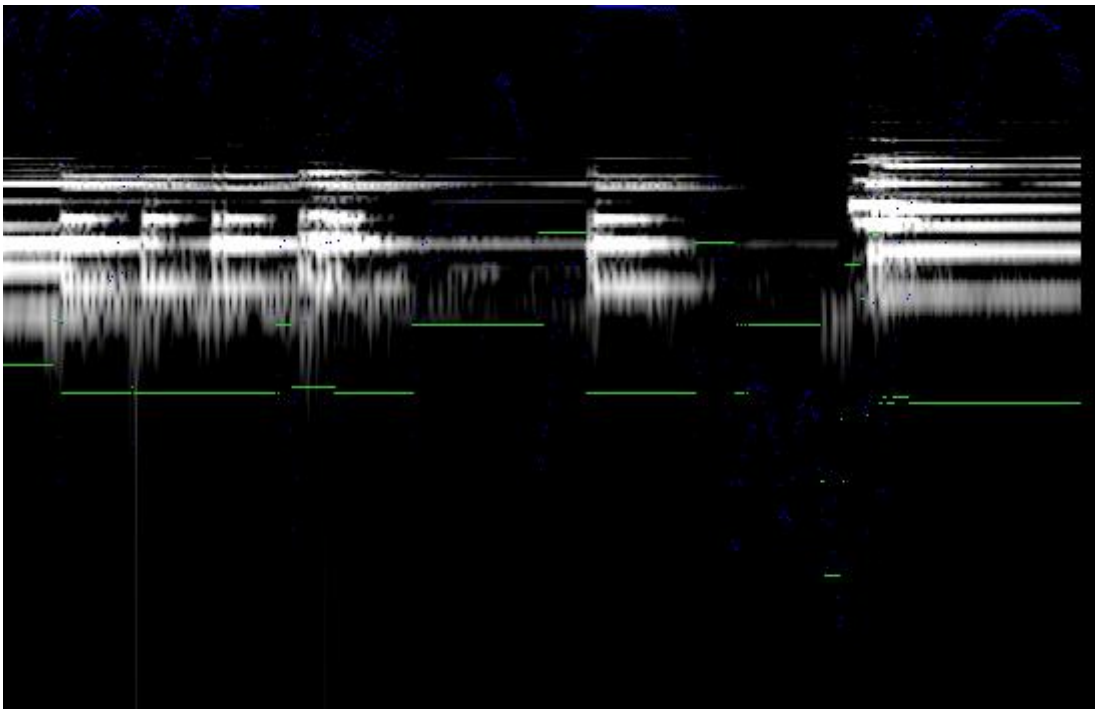
K uspokojivému výkonu sítě dochází i při použití téže skladby v MIDI, ale v dvojici s její syntetizovanou podobou vygenerovanou nástrojem TiMidity++ (volba „Test MIDI“ v aplikaci). Přestože tyto vzorky nebyly součástí trénovacích dat, síť na nich dosahuje průměrné přesnosti 59 %:



Obrázek 12: Ukázka výkonu druhé sítě na vzorové hudbě, ale syntetizované s výchozím soundfontem.

Takto vytvořenou síť lze do jisté míry použít i na reálné nahrávky piana, kde dokáže v mnoha případech identifikovat správné noty, ale rozsah jejího použití je stále

omezený. Především je v každém případě výsledný zvuk potřeba vyčistit a odstranit krátké mezery či deviace v delších notách, a jelikož (jak bylo ilustrováno) má tato síť horší rozpoznávací schopnost na nižších frekvencích, vstupní zvuk musí být upraven tak, aby se většina frekvencí nacházela v optimálním intervalu.



Obrázek 13: Ukázka výkonu druhé sítě na reálné hudbě (pop.ogg).

Závěr

Cílem práce bylo vytvořit aplikaci, kterou lze lehce používat pro pokusy aplikování neuronové sítě na hudbu. Přestože bylo ukázáno, že použití sítě učené pouze z malého vzorku generovaných dat není možné pro reálné nahrávky hudby či hudbu používající jiný nástroj či soundfont, i tak lze tyto sítě s uspokojivou úspěšností aplikovat v situacích, kdy máme jistou představu o vlastnostech hudby, tedy dopředu známe nástroje či rozsah not.

Také je nutno dodat, že používání čistého výstupu sítě bez postprocesu s tímto návrhem pravděpodobně nikdy nepovede k příliš vysoké úspěšnosti, především kvůli dříve zmíněným podstatným rozdílům mezi hudební reprezentací a skutečným zvukem. Podobně jako při řešení problému strojového rozpoznání řeči neuronovými sítěmi je obvykle potřeba výstup opravovat (srovnáním se slovníky), tak i zde by bylo pro opravení chyb v časování a trvání potřeba dalších nástrojů a procesů. Vzhledem k výstupu aplikace v MIDI lze však tyto nástroje vyvinout a doplnit externě.

Seznam použité literatury

BALHAR, Jiří. *Extrakce melodie pomocí hlubokého učení* [online]. Praha, 2019 [cit. 2020-03-17]. Dostupné z: https://www.researchgate.net/publication/252067543_Neural_networks_for_musical_chords_recognition. Bakalářská práce. Karlova univerzita, Matematicko-fyzikální fakulta, Ústav formální a aplikované lingvistiky. Vedoucí práce Mgr. Jan Hajič.

Category:Formats. *Video Game Music Preservation Foundation Wiki* [online]. 2019 [cit. 2020-03-17]. Dostupné z: http://www.vgmpf.com/Wiki/index.php?title=Category:Formats

HARRIS, F.J. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE* [online]. 1978, **66**(1), 51-83 [cit. 2020-07-27]. DOI: 10.1109/PROC.1978.10837. ISSN 0018-9219. Dostupné z: http://ieeexplore.ieee.org/document/1455106/

KIM, Jong Wook, Justin SALAMON, Peter LI a Juan Pablo BELLO. Crepe: A Convolutional Representation for Pitch Estimation. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* [online]. IEEE, 2018, 2018, s. 161-165 [cit. 2020-03-17]. DOI: 10.1109/ICASSP.2018.8461329. ISBN 978-1-5386-4658-8. Dostupné z: https://ieeexplore.ieee.org/document/8461329/

LEPIL, Oldřich. *Fyzika pro gymnázia*. 4. vyd. Praha: Prometheus, 2009. ISBN 978-80-7196-387-5.

MIDI MANUFACTURERS ASSOCIATION. General MIDI (GM 1) *MIDI.Org* [online]. [cit. 2020-03-17]. Dostupné z: https://www.midi.org/specifications-old/item/general-midi

MIDI MANUFACTURERS ASSOCIATION. GM 1 Sound Set *MIDI.Org* [online]. [cit. 2020-03-17]. Dostupné z: https://www.midi.org/specifications-old/item/gm-level-1-sound-set

OPPENHEIM, Alan V., Ronald W. SCHAFER a John R. BUCK. *Discrete-time signal processing*. 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 1999. ISBN 0-13-754920-2.

OSMALSKYJ, Julien, Jean Jacques EMBRECHTS, Marc VAN DROOGENBROECK a Sébastien PIÉRARD. *Neural networks for musical chords recognition* [online]. 2012 [cit. 2020-03-17]. Dostupné z: https://www.researchgate.net/publication/252067543_Neural_networks_for_musical_chords_recognition

VAN DEN OORD, Aaron, Sander DIELEMAN, Heiga ZEN, Karen SIMONYAN, Oriol VINYALS, Alex GRAVES, Nal KALCHBRENNER, Andrew SENIOR a Koray KAVUKCUOGLU. *WaveNet: A Generative Model for Raw Audio*. 2016 [cit. 2020-03-17].

VERMA, Dhruv. Music Transcription using a Convolutional Neural Network. *Medium* [online]. Dec 11, 2017 [cit. 2020-03-17]. Dostupné z: https://medium.com/@dhruvverma/music-transcription-using-a-convolutional-neural-network-b115968829f4

SHANNON, C.E. Communication in the Presence of Noise. *Proceedings of the IRE* [online]. 1949, **37**(1), 10-21 [cit. 2020-03-17]. DOI: 10.1109/JRPROC.1949.232969. ISSN 0096-8390. Dostupné z: http://ieeexplore.ieee.org/document/1697831/

Seznam obrázků

Seznam použitých zkratek

AAC	Advanced Audio Coding
CNTK	Cognitive Toolkit
CUDA	Compute Unified Device Architecture
DRO	DOSBox Raw OPL
FLAC	Free Lossless Audio Codec
FM	frekvenční modulace
iFFT	Inverse Fast Fourier Transform
IMF	id's Music Format
IT	Impulse Tracker
MAPS	MIDI Aligned Piano Sounds
MIDI	Musical Instrument Digital Interface
MNIST	Modified National Institute of Standards and Technology
MP3	MPEG-1 Audio Layer III
SMF	Standard MIDI File
SGD	Stochastic Gradient Descent
XM	eXtended Module
XML	eXtensible Markup Language

A. Přílohy

A.1 Zdrojové kódy aplikace

Pro kompilaci aplikace je nutno stáhnout všechny závislosti pomocí `nuget restore` a otevřít projekt pomocí nástroje Visual Studio (alespoň verze 2017) s podporou pro C# 7.3 a .NET Framework 4.7.2. Aplikace byla vytvořena a testována pouze na platformě Windows; pro její použití na Linuxu je potřeba použít prostředí Mono[18], nahradit nástroje ffmpeg a TiMidity++ jejich verzemi pro Linux a doplnit do systému nástroj `cmd`, který aplikace využívá pro propojení nástrojů ffmpeg a TiMidity++ (může být dodán mezivrstvou Wine[19]).

Ve složce MIDIO se nachází projekt aplikace (s adresáři rozdělenými podle jmenných prostorů popsaných v kapitole 4.3 Architektura aplikace). V adresáři tools se nacházejí nástroje ffmpeg a TiMidity++ (využívající soundfont Scc1t2[20]), které je nutno před spuštěním aplikace s celým adresářem překopírovat do výstupního adresáře. V adresáři examples jsou ukázkové projekty a v adresáři tests hudební soubory, oboje použito v kapitole 5. Evaluace.

[18] https://www.mono-project.com/

[19] https://www.winehq.org/

[20] https://stash.reaper.fm/v/23360/Scc1t2.sf2