VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ROZPOZNÁVÁNÍ ŽÁNRU POPULÁRNÍCH SKLADEB

RECOGNISING THE GENRE OF POPULAR SONGS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **FILIP ČIŽMÁR**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing FRANTIŠEK ZBOŘIL, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Čižmár Filip**
Program: Informační technologie
Název: **Rozpoznávání žánru populárních skladeb**
Recognising the Genre of Popular Songs
Kategorie: Umělá inteligence

Zadání:

1. Vytvořte kolekci trénovacích a testovacích dat pro systém, který má pro hudební skladby populární hudby rozpoznat jejich žánr (pop, disko, metal, dance, jazz ...)
2. Nastudujte a zvolte vhodné klasifikátory a způsoby jejich učení pro tuto úlohu.
3. Vytvořte aplikaci obsahující systémy zvolené v předchozím bodě, která jednak bude schopna trénovat klasifikátor a jednak ji bude možné používat pro přehrávání skladeb s tím, že bude odhadovat žánr aktuálně přehrávané skladby.
4. Vyhodnoťte fungování vytvořeného systému a navrhněte její možné další vylepšení do budoucna.

Literatura:

- K. Maheshkumar, Music Genre Recognition Using Deep Neural Networks and Transfer Learning, Interspeech.2018-2045, 2018
- Vaidya, K.: Music Genre Recognition using Convolutional Neural Networks, online [10.18.2021] https://towardsdatascience.com/music-genre-recognition-using-convolutional-neural-networks-cnn-part-1-212c6b93da76, 2020

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání

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

Vedoucí práce: **Zbořil František, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Cieľom práce je zoznámiť sa s princípmi práce so zvukom v programovacom jazyku Python a s problematikou konvolučných neurónových sietí s cieľom tvorby webovej aplikácie schopnej rozpoznávať žáner nahranej skladby. V práci sú opísané princípy strojového učenia so zameraním na konvolučné neurónové siete. Zásadná časť práce je venovaná prieskumu dostupných datasetov pre účely získavania informácií z hudby. Ďalej je opísaný priebeh prípravy vybraného datasetu a transformácie zvukovej informácie do podoby spektrogramov pre učenie konvolučnej neurónovej siete. V rámci práce boli vytvorené dva modely schopné rozpoznávať žáner hudby. Prvý z nich je všeobecný a druhý so zameraním na podžánre elektronickej hudby. Výsledkom je webová aplikácia, ktorá po nahraní skladby zobrazí pravdepodobnosti zaradenia do jednotlivých žánrov.

Abstract

The aim of this thesis is to get acquainted with the principles of working with sound in the Python programming language and with the issue of convolutional neural networks in order to create a web application capable of recognizing the genre of an uploaded song. The thesis describes the principles of machine learning with a focus on convolutional neural networks. A considerable part of this thesis is devoted to the research of available datasets created for the purpose of music information retrieval. Next, the process of preparation of the selected dataset and transformation of audio information into spectrograms for the learning of convolutional neural networks is described. Two models capable of recognizing the genre of music were created as a part of the thesis. First, for general, more popular genres and the second focuses on subgenres of electronic music. The result is a web application that, after a song is uploaded, displays the probabilities of classification into individual genres.

Kľúčové slová

Zvuk, hudba, žánre hudby, populárne skladby, elektronická hudba, Fourierove transformácie, spektrogramy, strojové učenie, neurónové siete, konvolučné neurónové siete.

Keywords

Sound, music, music genres, popular songs, electronic music, Fourier transforms, spectrograms, machine learning, neural networks, convolutional neural networks.

Citácia

ČIŽMÁR, Filip. *Rozpoznávání žánru populárních skladeb*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing František Zbořil, Ph.D.

Rozpoznávání žánru populárních skladeb

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením doc. Ing Františka Zbořila, Ph.D. a uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Filip Čižmár
10. mája 2022

Obsah

Kapitola 1

Úvod

Úloha zaradenia danej skladby do konkrétneho žánru je navonok zdanlivo jednoduchá. Každý vie, že skladba od skupiny The Rolling Stones bude pravdepodobne spadať pod rock. Ako tomu ale všeobecne býva, čím hlbšie sa človek zamýšľa nad jednou témou, tým komplikovanejšie otázky vychádzajú najavo. Úloha zaradenia do žánru tomuto nie je výnimkou. Existuje obrovské množstvo skupín a interpretov produkujúcich obrovské množstvo skladieb rôznych charakterov a z rôznych období. Tieto skladby niekedy bývajú konformné prvkom jedného populárneho žánru danej doby a sú ľahko zaraditeľné. V iných prípadoch si berú inšpiráciu z žánrov viacerých, prípadne žiadnych. Napriek tomu je ľudský mozog veľmi efektívny vo vyhľadávaní opakujúcich sa vzorov a príznakov a pre znalca danej hudby zaradiť takúto skladbu do jedného či viacerých žánrov určite nie je problém. Ako tomu je ale v prípade strojov? Tejto otázke sa venuje táto práca.

Cieľom je vytvoriť program schopný rozpoznávať žánre skladby. Na úlohy takéhoto charakteru sa zameriava disciplína strojového učenia a pre prácu je vhodné zamerať sa konkrétne na neurónové siete. Podobne ako ľudský mozog, neurónové siete sú schopné rozpoznávať opakujúce sa vzory a v kombinácií s určitými trénovacími dátami je možné neurónovú sieť naučiť rozpoznávať aj hudobné žánre. Z algoritmov neurónových sietí sa práca zameriava konkrétne na využitie sietí konvolučných. Strojové učenie všeobecne a konvolučné neurónové siete sú spracované v 2. sekcii 2. kapitoly tejto práce.

Pre tvorbu kvalitnej neurónovej siete sa predpokladá solídnosť datasetu využitého pri učení neurónovej siete. Z tohto dôvodu bola zásadná časť práce venovaná prieskumu jednotlivých riešení pre tvorbu či získanie vhodného datasetu. Nakoľko cieľom je využitie siete konvolučnej, takýto dataset musí obsahovať samotné audio súbory skladieb vrátane ich roztriedenia do žánrov. Ostatné skutočnosti týkajúce sa problémov a výberu datasetu sú opísané v 3. kapitole. Postupy preprocessingu a spôsoby transformácie audio dát do vizuálnej podoby pre učenie neurónovej siete sú spracované vo 4. kapitole práce.

V rámci práce boli celkovo vytvorené dve neurónové siete. Prvá bola naučená na dátach obsahujúcich žánre všeobecné. Pre druhú neurónovú sieť boli z datasetu extrahované skladby podžánrov elektronickej hudby a model bol učený na týchto skladbách. Obidva modely boli následne využité pri tvorbe webovej aplikácie. Procesy učenia siete a tvorby webovej aplikácie sú opísané v kapitolách 5 a 6.

Kapitola 2

Súčasný stav

Táto kapitola sa venuje popisu teoretického aspektu problematiky, počínajúc zvukom. Pomerná časť kapitoly je venovaná spôsobu spracovania zvukových dát a klasifikácie pomocou neurónových sietí. Na záver sú načrtnuté technológie pre tvorbu webových aplikácií.

2.1 Popis zvuku

Zvuk je mechanické vlnenie, ktoré sa šíri pružným prostredím (plyny, kvapaliny, pevné látky) schopné vyvolať v ľudskom uchu sluchový vnem [29]. V záujme zjednodušenia definície môžeme povedať, že ide o kmitanie molekúl danej látky – tieto molekuly rozkmitajú susedné (týmto spôsobom sa zvuk šíri, vzniká zvuková vlna). Zároveň je to dôvod prečo zvuk neexistuje vo vákuu. Zvuk má mnoho vlastností, podľa ktorých sa tiež rozdeľuje (k tomuto viac neskôr), ale každý zvuk má spoločnú podobnosť stavby. Všetky zvuky sú fundamentálne tvorené skladaním sínusoíd.

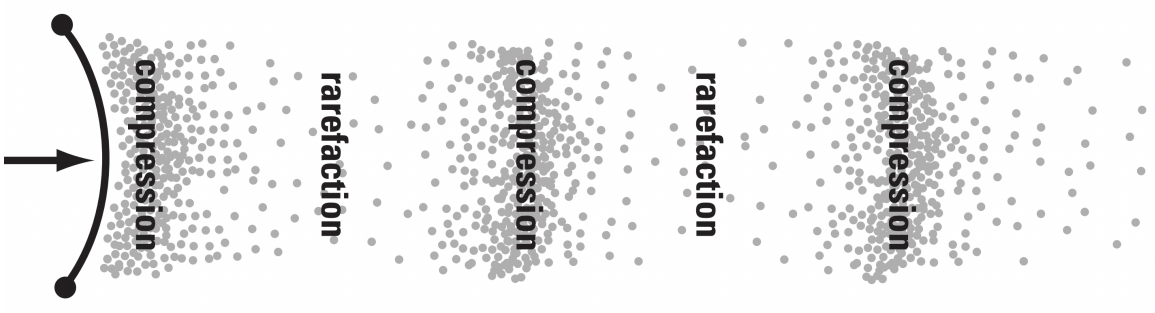
Spôsob šírenia zvuku

[17] Zvukové vlny vznikajú periodickým javom stláčania molekúl k sebe a následným odpružením od seba. Predstavme si zvuk prasknutia balóna. Nafúknutý balón obsahuje obrovské množstvo kompresovaných častíc vzduchu. Prasknutím balónu sa tieto častice veľkou rýchlosťou rozletia všetkými smermi a narazia na susedné molekuly. Vzniká vlna vysokého tlaku, ktorú nazývame *zhustenie* (*compression*). Na mieste, kde sa častice pôvodne nachádzali vzniká vlna nízkeho tlaku, ktorú nazývame *zriedenie* (*rarefaction*). Častice vzduchu následne tento priestor znova vyplnia. Proces sa opakuje periodicky dokým sa tlak na obidvoch stranách nevyrovná. 2.1

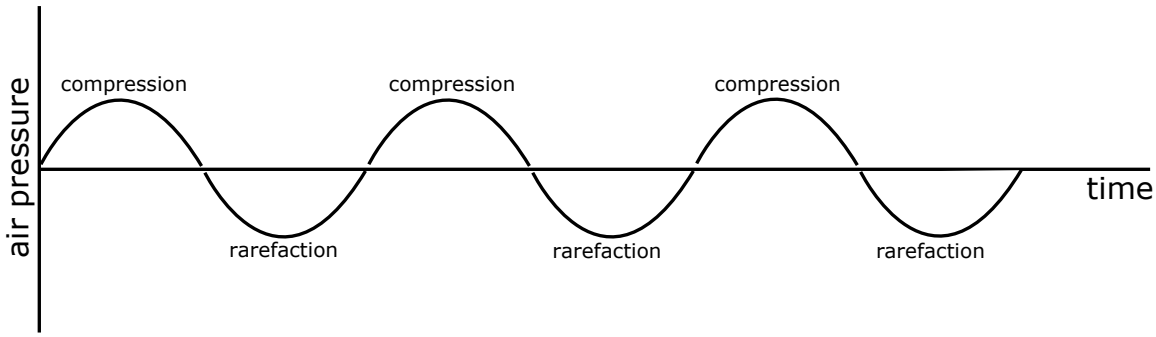
Zachytenie zvuku

Zachytenie zvuku prebieha v jednom bode v priestore premietnutím tlaku častíc (zhustení a zriedení) na čas. 2.2 Spôsob akým počuje človek je, podobne ako u ostatných častí ľudského teľa, komplikovaný mechanizmus. Pre účely tejto práce stačí poznamenať nasledujúce fakty: [3] 2.3

1. Zvuková vlna je smerovaná ušným boltcom (pinna) do vonkajšieho zvukovodu (auditory canal).



Obr. 2.1: Vizualizácia zhustenia a zriedenia. Zdroj: [17]

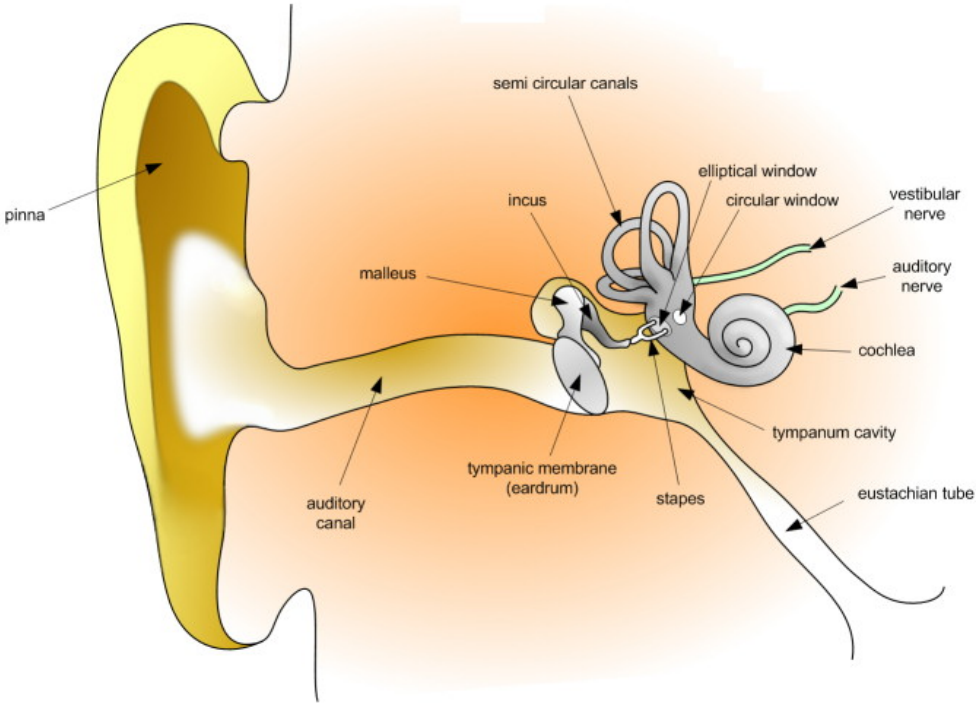


Obr. 2.2: Premietnutie zhustenia a zriedenia do sínusoidy.

2. Zvukovod privádza zachytené zvuky k bubienku (tympanic membrane), ktoré do neho narážajú a rozochvievajú ho.
3. Akustický tlak sa cez bubienok, kladivko (malleus), nákovku (incus) a strmienok (stapes) dostáva k oválnemu okienku. To tvorí hranicu medzi stredným a vnútorným uchom.
4. Rozkmitanie oválneho okienka spôsobí vibrácie v endolymfe (nestlačiteľná kvapalina), ktorá prostredníctvom membrana tectoria dráždi vláskové bunky.
5. Tie uvoľňujú malé množstvo mediátoru, čím vzniká **nervový signál**.

Vlastnosti zvuku

- Amplitúda – veľkosť tlaku v mieste maxima. Určuje intenzitu zvuku.
- Intenzita zvuku – udáva tok zvukovej energie. S rastúcim akustickým tlakom rastie aj intenzita a hlasitosť.
- Hlasitosť – subjektívne hodnotená intenzita zvuku. Pomer hlasitosti zvuku k určitej prahovej úrovni hlasitosti je udávaný v *decibeloch* (dB). Dynamický rozsah ľudského ucha, teda rozsah počuteľný ľudským uchom je *0 – 140 dB*, kde okolo 130dB nastáva tzv. prah bolestivosti.
- Frekvencia – počet zvukových vĺn prechádzajúcich miestom merania za časovú jednotku (obecne 1s). Udáva sa v *hertzoch* (Hz).



Obr. 2.3: Znázornenie sluchového ústrojenstva. Zdroj: [3]

Digitalizácia zvuku

Podobne ako u ľudského ucha, nahrávanie zvuku zahŕňa zachytenie vzoriek vibrácií vo vzduchu a ich konverziu na analógový elektrický signál. Z analógového elektrického signálu následne vzniká digitálny.

1. Zvukové vlny sú zachytené *mikrofónom*. [34] Mikrofón obsahuje magnet a membránu, ktorá sa nárazom častíc rozkmitá. K membráne je pripevnená cievka, ktorá sa hýbe s ňou. Pohybom cievky cez magnetické pole vzniká elektrický prúd.[1]

2. Elektrická energia sa z mikrofónu presúva káblom a pretože energia vytvorená takýmto spôsobom je väčšinu času pomerne malá – používa sa predzosilňovač.

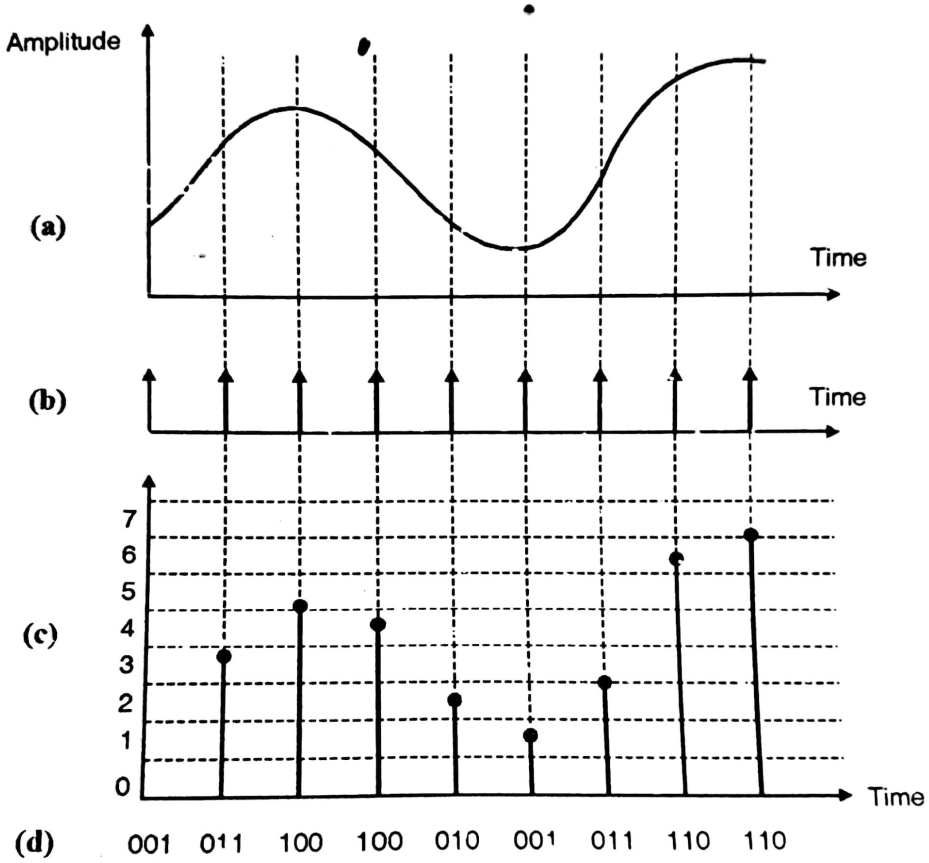
3. Na premenu analógového elektrického signálu na digitálny (reťaz binárnych čísel) sa používa zariadenie zvané ADC (*Analog to Digital Converter*).

Vzorkovanie (Sampling)

[1] Analógový signál je meraný A/D (ADC) prevodníkom v *diskrétnych* časových intervaloch. Výsledkom je hodnota napätia z mikrofónu v danom čase. Časová os sa delí do časových intervalov podľa **vzorkovacej frekvencie**.

Vzorkovacia frekvencia (sampling rate) stanovuje počet meraných vzoriek za sekundu. Frekvencia 48kHz hovorí, že hodnota signálu bude zaznamenaná 48 000 krát za sekundu. Vyššia vzorkovacia frekvencia = kvalitnejší zvuk.

[1] Nie všetky mikrofóny fungujú presne takýmto spôsobom. Princíp je ale podobný.



Obr. 2.4: Postup digitalizácie zvuku. a)spojitý signál, b)vzorkovanie, c)kvantovanie d)kódovanie

Elegantnejšie povedané, v teórii signálov ide o diskretizáciu v časovej oblasti, prípadne redukciu spojitého času na diskrétny [6].

Existuje tzv. *Nyquistov teorém*, ktorý tvrdí, že vzorkovacia frekvencia musí byť aspoň dvojnásobná oproti frekvencii pôvodného signálu. Z tohto dôvodu, ak priemerný človek počuje zvuk do 20kHz, sa najčastejšie používa vzorkovacia frekvencia nad 40kHz. V praxi s 10% navýšením preto najčastejšie vidíme 44,1kHz a 48kHz. V profesionálnych nahrávkach a zariadeniach je táto frekvencia omnoho vyššia.

Kvantovanie (Quantizing)

Zakaždým, keď ADC vykoná meranie, musí vzorke priradiť nejakú hodnotu. Tento proces sa nazýva kvantovanie. Spojité (nekonečné / príliš veľké) množstvo hladín signálov je obmedzené na diskrétne intervaly hodnôt. Zanedbanie krajných hodnôt a zaokrúhlenie býva často typickým dôsledkom procesu kvantovania.

[1] Zdanlivo ide o moduláciu (konvolúciu) pôvodného signálu signálom s pravidelnou pulznou charakteristikou, z čoho pochádza aj označenie výsledného číselného zápisu takto získaného zvuku – PCM (pulse-code modulation, teda pulzne kódová modulácia).

Kódovanie

V digitálnych systémoch je hodnota danej vzorky zaznamenaná binárnym číslom. Dĺžka binárneho čísla je v tomto prípade **rozlíšenie vzorky** alebo **bitová hĺbka**. Bitová hĺbka 16 teda značí 65 536 úrovní signálu.

Zvukové formáty

Audio formát alebo formát kompresie audia označuje spôsob reprezentácie audia pre digitálne nosiče a prenos. Napriek tomu, že zvukových formátov existuje nespočetne, pre účely tejto práce stačí spomenúť len niektoré, menovite hlavne PCM (.wav) a MP3.

Existujú stratové (lossy), bezstratové (lossless) a nekomprimované formáty (uncompressed) [2].

1. **Bezstratové** formáty kódovania zvuku znižujú celkové údaje potrebné na reprezentáciu zvuku, ale možno ich dekódovať do pôvodnej, nekomprimovanej podoby.

   - FLAC (Free Lossless Audio Codec)
   - ALAC (Apple Lossless Audio Codec)

2. **Stratové** formáty kódovania zvuku navyše znižujú bitové rozlíšenie zvuku nad rámec kompresie, čo vedie k oveľa menšiemu počtu údajov za cenu nenávratne stratených informácií.

   - MP3 – používa aproximáciu a čiastočné vyraďovanie dát na zníženie celkových dát až o 75 – 95% [5] oproti audiu CD kvality (16-bit, 44,1 kHz)

3. **Nekomprimované** formáty sa vyznačujú najlepšou kvalitou a úrovňou zachovania informácií (ukladajú pôvodné audio). Nevýhoda spočíva v značnej veľkosti dát. Vypočítanie veľkosti audia takéhoto formátu je triviálna záležitosť.

   Pri vzorkovacej frekvencii 44,1kHz, bitovej hĺbke 16 bitov má jeden kanál veľkosť 88,2kB/s – približne 5,3MB za minútu na mono audio alebo 10,6MB na stereo. HD audio (24-bit, 192kHz, stereo) môže zaberať až 69MB na minútu audia.

   - .WAV (PCM) – Waveform Audio File Format – kódovanie bitového toku pomocou formátu LPCM. Najčastejšie používaný formát pre nekomprimované audio.

2.2 Spektrálna analýza

Fourierova transformácia

Fourierova transformácia je jedným zo základných nástrojov pre prácu so signálmi. V zásade slúži na transformáciu signálu z časovej oblasti do oblasti frekvenčnej. V spojitom čase je Fourierova transformácia $S(\omega)$ pre signál $s(t)$ definovaná nasledovne [4]:

$$S(i\omega) = \int_{-\infty}^{\infty} s(t)e^{-i\omega t}dt \tag{2.1}$$

$S(\omega)$ predstavuje spektrum signálu $s(t)$.

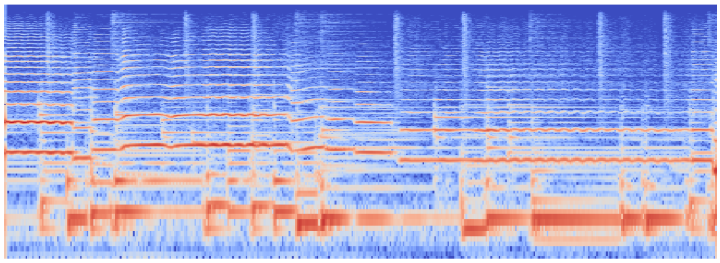
Nakoľko ale stroje a algoritmy nepracujú v čase spojitom, ale naopak, v čase diskrétnom, pre účely tejto práce je zaujímavejšia **Diskrétna Fourierova Transformácia (DFT)**. Tá môže byť matematicky zapísaná nasledovne [20]:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-i2\pi \frac{k}{N} n} \tag{2.2}$$

Diksrétna Fourierova transformácia transformuje postupnosť N komplexných čísiel $x_n = \{x_0, x_1, ..., x_{N-1}\}$ na inú postupnosť komplexných čísiel $X_n = \{X_0, X_1, ..., X_{N-1}\}$

Spektrogramy

Zjednodušene povedané, spektrogram je jedna z možných transformácií audia do vizuálnej formy. Zobrazuje čas, frekvenciu a amplitúdu na jednom grafe. Na rozdiel od tradičného zobrazenia signálu (*waveform*), ktorý zobrazuje amplitúdu za čas, spektrogram zobrazuje frekvenciu na vertikálnej ose a amplitúdu zafarbením (svetlosťou). Príklad spektrogramu je možné vidieť na obr. 2.5.



Obr. 2.5: Príklad spektrogramu jazzovej skladby

Prevod audia do spektrogramu je stratový, nakoľko nezachováva všetky informácie z pôvodnej skladby. Napriek tomu je audio rozpoznateľné a využitie spektrogramu je vhodné v mnohých úlohách zvukovej analýzy.

Spektrogram je možné vytvoriť priamym využitím Fourierovej transformácie na krátke úseky audia nasledujúce po sebe. Transformácia v takejto podobe sa nazýva krátkodobá Fourierova transformácia, alebo **STFT** (Short Term Fourier Transform).

Matematický vzorec krátkodobej Fourierovej transformácie môže vyzerať napr. takto:

$$X(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t - \tau) e^{-i\omega t} dt \tag{2.3}$$

Pre spojitý signál $x(t)$ je možné vypočítať STFT podľa vzorca 2.3, kde $w(\tau)$ je zvolené okno. **Upozorňujem na podobnosť medzi ω (omega) a w (písmeno w)**, kde ω je frekvencia a w oknová funkcia.

Signál získaný z audio záznamu hudby ale obsahuje diskrétny čas vo forme vzorkovacej frekvencie. Z tohto dôvodu je nutné využitie diskrétnej krátkodobej Fourierovej transformácie. V porovnaní s verziou v spojitom čase dochádza k niektorým zmenám.

$$X(m,\omega) = \sum_{n=-\infty}^{\infty} x(n)w(n-mR)e^{-j\omega n} \tag{2.4}$$

Výpočet diskrétnej STFT je možný podľa vzorca 2.4, kde $x(n)$ je vstupný signál v diskrétnom čase n, $w(n)$ je oknová funkcia (napr. Hammingova) a R je veľkosť skoku vo vzorkách medzi jednotlivými DFT.

Mel spektrogram

Osobitným typom spektrogramov sú mel spektrogramy. Tieto spektrogramy využívajú logaritmickú melovskú stupnicu na prevedenie frekvenčných hladín na hodnoty bližšie k tomu, čo počuje ľudské ucho podľa vzťahu vo vzorci 2.5.

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.5}$$

Výsledkom využitia melovskej stupnice sú melovské spektrogramy, ktorých zásadnou výhodou je lepšie využitý priestor frekvenčných hladín počuteľných ľudským uchom. Nakoľko hudba je primárne tvorená pre ľudské počúvanie, žiadané frekvencie takýchto zvukov sa pohybujú práve v týchto hladinách.

2.3 Strojové učenie a neurónové siete

Strojové učenie je odvetvie zaraďujúce sa pod úlohy umelej inteligencie. Základným znakom strojového učenia je imitácia ľudskej schopnosti získavať poznatky a na ich základe sa *učiť*, zvyšujúc tak presnosť a kvalitu svojich výsledkov.

Postupne sa stalo dôležitou súčasťou vedy o dátach (angl. "data science"). Pomocou štatistických metód sú algoritmy trénované na vytváranie klasifikácií alebo predpovedí, ktoré odhaľujú kľúčové poznatky v rámci projektov dolovania údajov [14].

Medzi najčastejšie používané algoritmy strojového učenia patria napr. rozhodovacie stromy, SVM (Support vector machines), K-Means clustering.

Strojové učenie je možné rozdeliť do 3 skupín [36]:

1. **Učenie s učiteľom (supervised learning)** – spočíva v tom, že pre každý krok učenia je známa požadovaná odozva a systém je tak okamžite informovaný o aktuálnom hodnotení jeho poslednej akcie.

   Učenie prebieha na trénovacej množine príkladov **T**, kde každý príklad je tvorený množinou vstupných hodnôt (vstupným vektorom) a množinou požadovaných výstupných hodnôt (výstupným vektorom):

$$T = \{(\vec{i}_1, \vec{d}_1), (\vec{i}_2, \vec{d}_2), ..., (\vec{i}_p, \vec{d}_p)\}$$

$\vec{i_i}$ ... i–tý vstupný vektor

$\vec{d_i}$ ... i–tý výstupný vektor

Celá množina dostupných dát je rozdelená na podmnožiny s požadovaným pomerom (najčastejšie 80%/20%), kde prvá (väčšia) podmnožina sa stáva trénovacou, na ktorej sa model učí, a druhá testovacou, na ktorej sa overuje kvalita doposiaľ naučeného modelu.

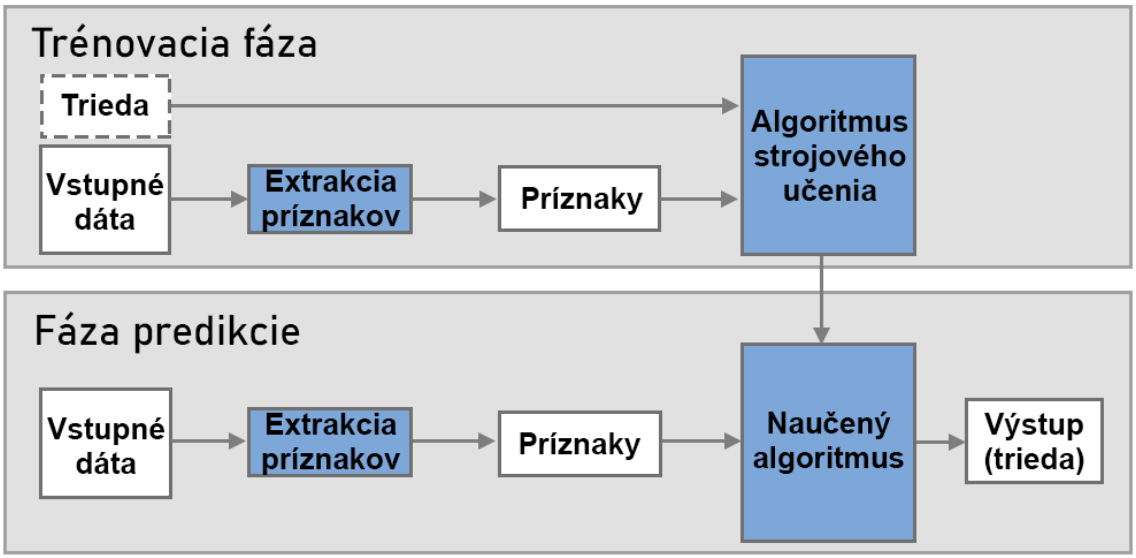
2. **Učenie bez učiteľa (unsupervised learning)** – využitie pri úlohách, kde množina požadovaných výstupných hodnôt nie je dostupná. Učenie teda prebieha nachádzaním podobností vo vstupných vektoroch trénovacej množiny. Trénovacia množina **T** je tvorená len vstupným vektorom:

$$T = \{(\vec{i_1}, \vec{i_2}, ..., \vec{i_p})\}$$

3. **Posilované učenie (reinforcement learning)** – posilované učenie je zdanlivo podobné učením s učiteľom, zásadne sa ale odlišuje v okolnostiach, pri ktorých je systém ohodnocovaný. Obvykle je ohodnotený až po dokončení nejakého zhluku postupných akcií. Toto ohodnotenie je rôzne veľké a môže byť kladné alebo záporné. Podľa tohto ohodnotenia systém sám zaistí zmeny.

Klasifikácia algoritmom strojového učenia prebieha obecne vo dvoch fázach [25]:

1. Trénovanie – učenie modelu na dostupných dátach opísané vyššie.
2. Predikcia / klasifikácia – využitie naučeného modelu na klasifikáciu nových vstupných dát.



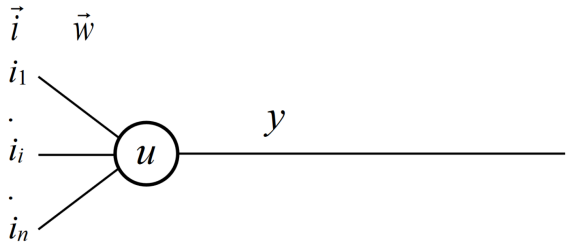
Obr. 2.6: Fázy strojového učenia (preložené z [25])

Na Obr. 2.6 je znázornený priebeh trénovacej fázy a predikcie vstupných dát. V trénovacej fáze sa model učí z príznakov extrahovaných zo vstupných dát. Po priebehu trénovacej fázy je výsledkom naučený algoritmus strojového učenia, ktorý je následne využitý na klasifikáciu nových vstupných dát. Pri klasifikácii nových dát znova prebehne proces extrakcie príznakov.

Neurónové siete

Neurónové siete spadajú pod algoritmy učenia s učiteľom (príp. učenie bez učiteľa) a ich názov aj princíp si povrchne berú inšpiráciu z fungovania neurónov v mozgu. Základným stavebným prvkom mozgu je neurón, ten sa následne skladá z tela, sto až stotisíc výbežkov zvaných dendrity a jedného výstupu zvaného axon. Rozhranie medzi dendritmi a axonmi jednotlivých rozdielnych neurónov sa nazýva *synapsia*. Tá sprostredkováva vplyv jedného neurónu na druhý. V krátkosti, všetko učenie, ktoré v mozgu prebieha, je založené na zmene synaptických váh (možné predstaviť si ako zmenu úrovne vplyvu jedného neurónu na druhý) [36].

Podobne funguje princíp algoritmu neurónovej siete, kde každý neurón môže mať mnoho vstupov ale len jeden výstup. Tento výstup môže byť prijatý viacerými neurónmi ako jeden z ich vstupov. Vstupom taktiež môže byť aj iná vonkajšia informácia (napr. namerané parametre nejakého stroju), nie len výstup iného neurónu. Analogicky ku synaptickým váham v mozgu, každý zo vstupov neurónu v neurónovej sieti má určitú váhu [13].



Obr. 2.7: Klasický model neurónu (zdroj: [36])

$$y = g(u) = g(f(\vec{i}))$$

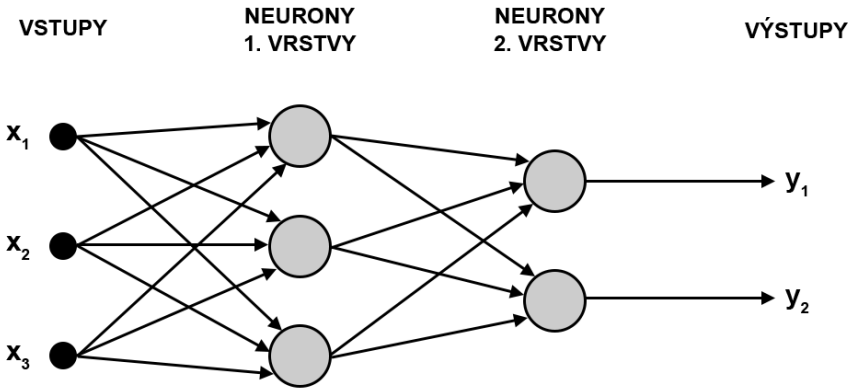
Na obr. 2.8 je znázornený model neurónu, kde $\vec{i}$ predstavuje vektor vstupov, $\vec{w}$ vektor váh (synapsií), u vnútorný potenciál neurónu a y je výstup neurónu opísaný bázovou (f) a aktivačnou (g) funkciou.

Bázové funkcie môžu byť lineárne alebo radiálne, aktivačné funkcie môžu byť skokové alebo spojité. Najjednoduchší umelý neurón je neurón s lineárnou bázovou a skokovou aktivačnou funkciou a nazýva sa perceptron.

$$y = g\left(\sum_{j=1}^{N} w_j \cdot i_j - \theta\right) \tag{2.6}$$

Vstupy neurónu sú vynásobené váhami. Ak je súčet všetkých vážených vstupov väčší než stanovený prah θ, výsledok je transformovaný vopred danou prenosovou funkciou a poslaný na výstup [13].

Podobne ako v mozgu, učenie neurónovej siete prebieha zmenou vstupných váh daného neurónu a zmenou prahu θ.



Obr. 2.8: Príklad dvojvrstvovej neurónovej siete (zdroj: [19])

Na obr. 2.7 je znázornený príklad perceptronovej neurónovej siete, kde prvá (vstupná) vrstva opakuje hodnoty troch vstupov a predá ich všetkým neurónom v nasledujúcej vrstve. Výstupy (celkom dva) z neurónov v druhej (výstupnej) vrstve určujú výstup celej neurónovej siete. Medzi vstupnou a výstupnou vrstvou bývajú obecne tzv. vrstvy *skryté*, sieť v príklade vyššie ich ale neobsahuje.

U jednovrstvových perceptronových sietí sa neuróny navzájom neovplyvňujú, každý neurón sa učí samostatne.

Pri učení viacvrstvových dopredných sietí sa využíva algoritmus **backpropagation** (spätné šírenie chyby), nakoľko dopredné siete nemajú v spätné väzby. Algoritmus spočíva v rozdelení výslednej chyby z predošlej iterácie medzi jednotlivé neuróny a následnej úprave váh podľa tohto rozdelenia.

Takouto neurónovou sieťou je napríklad CNN (konvolučná neurónová sieť).

2.4 Konvolučné neurónové siete

Konvolučné neurónové siete, skratkou CNN z anglického Convolutional Neural Networks sú špeciálnym typom dopredných viacvrstvových neurónových sietí navrhnuté pre emuláciu zrakovej kôry mozgu. Konvolučnú sieť je možné si predstaviť ako neurónovú sieť so špecializáciou v nachádzaní opakujúcich sa vzorov a častí vstupných dát. Vstupom tento raz nie je príznakový vektor, ale obrazy v ich pôvodnej podobe.

Rozšírenie konvolučnej neurónovej siete začalo v roku 2012, kedy metóda pod názvom Alexnet založená na CNN predbehla v každoročnej súťaži *Imagenet Visual Recognition Challenge* druhú najlepšiu metódu o takmer 11% [26].

Od vtedy sa konvolučné siete stali dôležitým nástrojom počítačového videnia a využívajú sa pri mnohých rozpoznávacích (angl. pattern recognition) úlohách, nielen na spracovanie

obrazu (rozpoznanie tváre, autonómne vozidlá), ale aj pri spracovaní reči, hudby, textu a podobne [26].

Najčastejšie využitie pre konvolučné siete je napriek tomu klasifikácia obrazu. V zásade je v tomto prípade na vstupe obraz a na výstupe trieda (napr. mačka, pes a pod.) alebo pravdepodobnosti tried, ktoré najlepšie opisujú obrázok. Keď sa pozrieme na obrázok, ľudský mozog vie veľmi jednoducho rozpoznať a zatriediť, čo na obrázku vidí na základe predošlých znalostí, aj keď je rozpoznávaný cieľ zobrazený v atypických okolnostiach. Toto je vlastnosť, ktorú s nami počítače nezdieľajú, a ktorá je predmetom riešenia konvolučných sietí [12].

Vrstvy siete a architektúra

Obr. 2.9: Príklad vrstiev konvolučnej neurónovej siete (zdroj: [26])

Ako je možné vidieť na obr. 2.9, konvolučná neurónová sieť sa skladá z rôznych typov vrstiev, z ktorých každý spĺňa iný druh úlohy. Architektúra konkrétnej konvolučnej siete potom záleží na účele využitia a podľa tohto cieľu sa menia aj jednotlivé použité vrstvy. Napr. využitie klasifikačnej vrstvy je požadované ak výstupom siete má byť trieda alebo vektor pravdepodobností tried.

Konvolučná vrstva

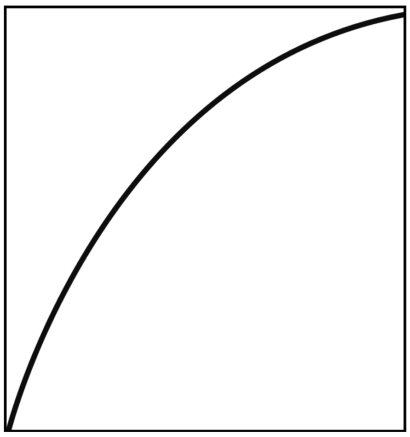
Výraz konvolúcia v CNN je odkaz na matematický operátor konvolúcie. Ide o lineárnu operáciu, ktorá spracováva dve funkcie a výsledkom je nová, tretia, ktorá opisuje ako sa vstupné 2 funkcie navzájom ovplyvňujú.

Z pohľadu CNN je vstupom konvolúcie obraz a nejaký **filter** (príp. *maska* alebo *kernel*) a výstup je využitý pri extrakcii príznakov z obrazu. Filter je možné si predstaviť ako lampu, ktorá svieti vždy na časť obrazu (*receptorové pole*) a postupne sa posúva až neprejde celý obraz.

V skutočnosti je filter pole o veľkosti W x H x C, kde W je šírka filtra, H jeho výška a C hĺbka. Hĺbka je určená počtom kanálov, jeden pre čiernobiele, tri kanály pre RGB. Hĺbka filtra musí byť rovná hĺbke vstupného obrazu. Pole filtra je naplnené váhami, ktoré určujú, na aký príznak má byť filter citlivý.

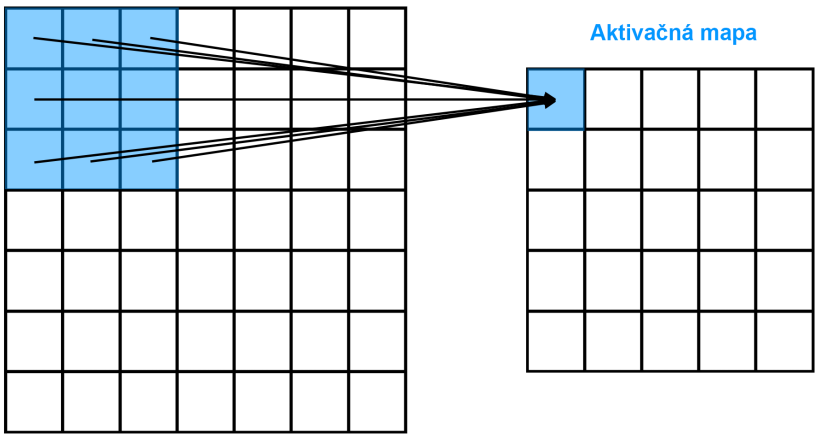
Na obr. 2.10 je znázornené pole 7x7x1 filtra, ktorý rozpoznáva prítomnosť krivky. Pri priebehu tohto filtra je prvá pozícia vľavo hore vo vstupnom obraze. Postupne ako sa filter posúva (*konvoluje*) po vstupe, násobí svoje hodnoty pola so zodpovedajúcimi hodnotami obrazu. Tieto násobené hodnoty sa sčítajú a výsledkom je jedno číslo, suma. Jedno číslo zodpovedá jednej pozícii filtra a posúvaním filtra po celom vstupe postupne vzniká pole

0	0	0	0	0	30	30
0	0	0	30	30	0	0
0	0	30	0	0	0	0
0	30	0	0	0	0	0
0	30	0	0	0	0	0
30	0	0	0	0	0	0
30	0	0	0	0	0	0



Obr. 2.10: Príklad zjednodušeného filtra detekujúceho krivku (upravené z: [12])

zvané **aktivačná** alebo **príznaková mapa**. Jednotlivé receptorové polia v individuálnych krokoch sa môžu navzájom prekrývať. Proces je znázornený na obr. 2.11



Obr. 2.11: Konvolúcia 3x3 filtru (upravené z: [12])

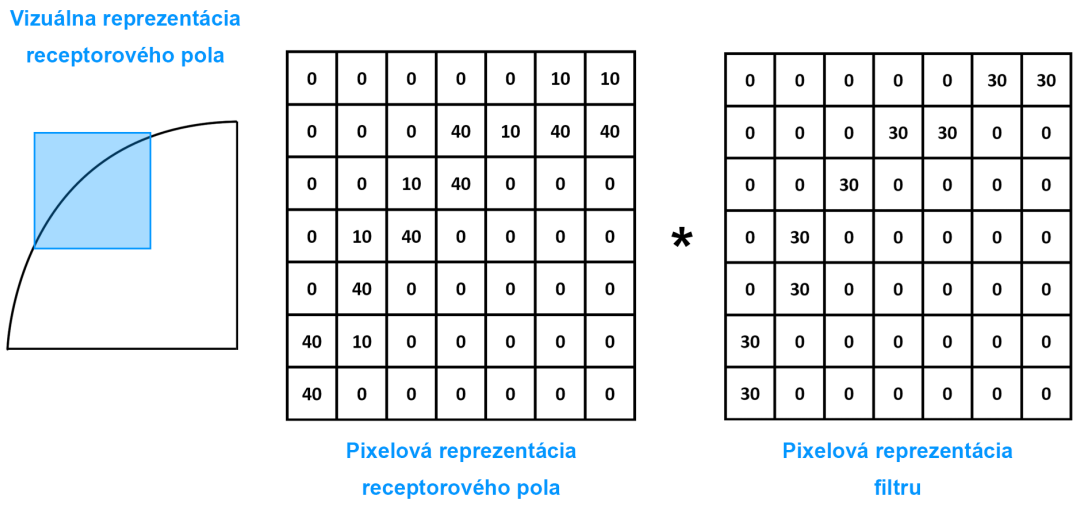
Ako je možné vidieť na obr. 2.12 a obr. 2.13, keď filter narazí na hodnoty, ktoré sedia s pozíciami v poli filtra, výsledná sčítaná hodnota po násobení bude porovnateľne väčšia než v iných (nepríznakových) pozíciách.

Výsledný neurón v aktivačnej mape podľa obr. 2.12 by mal nasledovnú hodnotu:

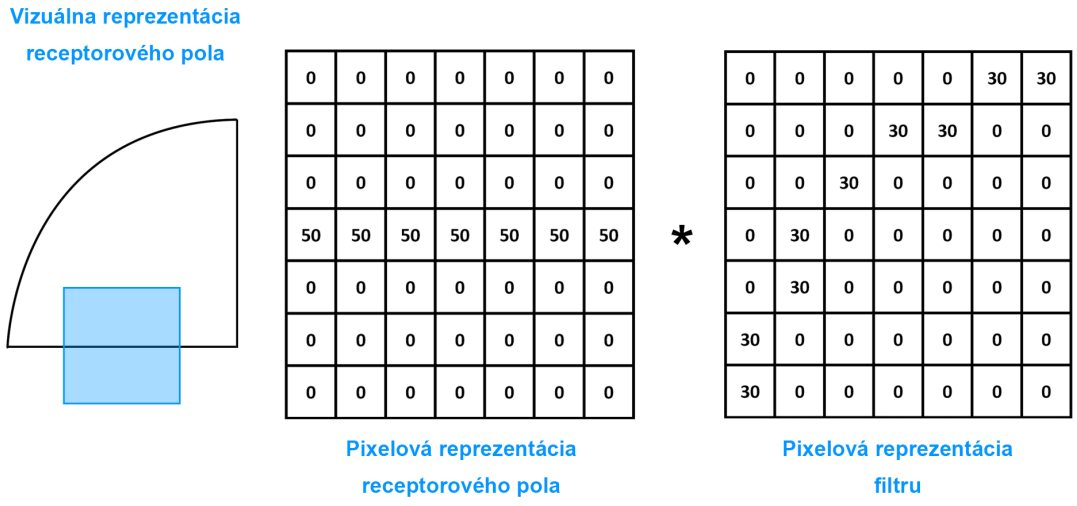
$$30 * 40 + 30 * 40 + 30 * 40 + 30 * 10 + 30 * 10 + 30 * 40 + 30 * 10 + 30 * 10 + 30 * 10 = 6300$$

Výsledný neurón v aktivačnej mape podľa obr. 2.13 by mal v tomto prípade hodnotu takúto:

$$30 * 0 + 30 * 0 + 30 * 0 + 30 * 50 + 30 * 0 + 30 * 0 + 30 * 0 + 30 * 0 + 30 * 0 = 1500$$



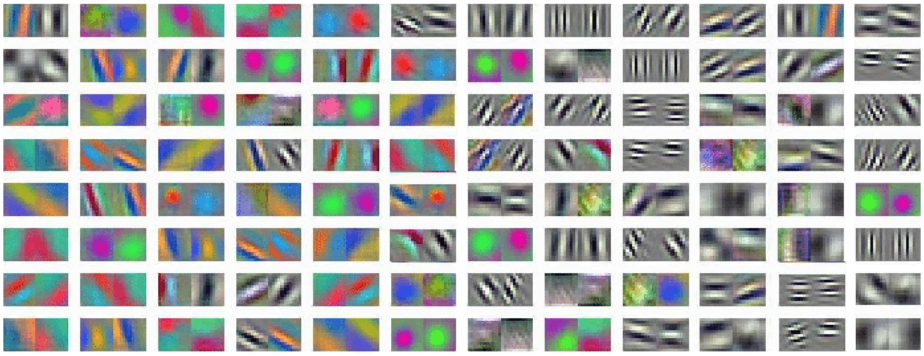
Obr. 2.12: Reprezentácia nájdeného príznaku (upravené z: [12])



Obr. 2.13: Reprezentácia nenájdeného príznaku (upravené z: [12])

Výsledkom priebehu filtra po vstupnom obraze je teda aktivačná mapa znázorňujúca pozície vstupného obrazu, na ktorých je vysoká resp. nízka pravdepodobnosť nálezu príznaku, na ktorý je filter citlivý.

Je dôležité poznamenať, že v praxi naberajú filtre omnoho komplikovanejších tvarov, než je napr. krivka znázornená na obr. 2.10. Príklady vizualizácií filtrov z konvolučnej neurónovej siete AlexNet je možné vidieť na obr. 2.14.



Obr. 2.14: Príklady vizualizácií filtrov siete AlexNet (zdroj: [31])

Pooling vrstva

Vo väčšine prípadov po konvolučnej vrstve nasleduje pooling (podvzorkovacia) vrstva. Základným cieľom podvzorkovania je zjednodušiť príznakovú mapu za účelom zmenšenia výpočtových nákladov. Princíp spočíva v združovaní oblastí statickej veľkosti príznakovej mapy – tým pádom sa zníži počet spojov medzi vrstvami. Čím väčšia je veľkosť vstupných častí, tým väčšej redukcie vo výstupných dátach sa dosiahne. Tento proces prebieha individuálne pre každú príznakovú mapu. Procesom podvzorkovania sa presná informácia a umiestení príznakov redukuje na približnú. Tento fakt ale nie je veľké negatívum, keďže pre účely neurónovej siete je relatívna lokalita príznaku viac než dostačujúca a vedieť presnú lokalitu nie je žiaduce. Ďalšou výhodou pooling vrstvy je prirodzené zníženie **overfittingu**. Overfitting je problém, ktorý nastáva keď je model príliš naladený na trénovaciu množinu údajov a aj napriek dosiahnutiu 99% presnosti na tejto množine, na množine testovacích alebo iných vstupných dát dosiahne menej než prívetivé výsledky [16][12].

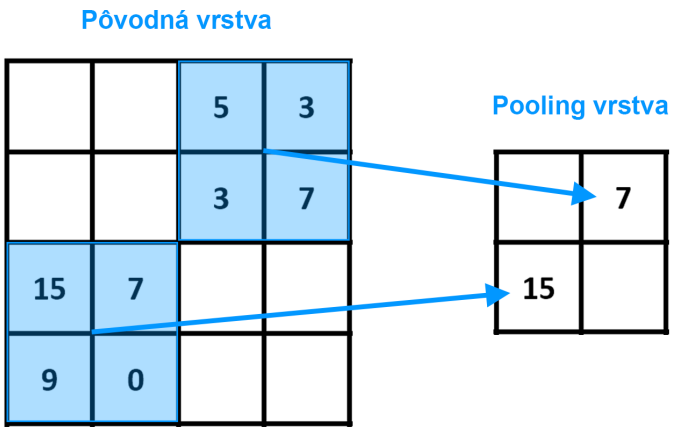
Podľa použitej metódy rozdeľujeme viacero druhov pooling vrstvy. Spravidla býva združovanie vykonané priemerom alebo maximom, ale existuje napr. aj združovanie súčtom.

- Združovanie maximom (Max Pooling) – z každej vstupnej oblasti pôvodnej vrstvy je prevzatý prvok s najväčšou hodnotou.
- Združovanie priemerom (Average Pooling) – výsledkom je priemer všetkých hodnôt prvkov vstupnej oblasti.

Príklad združovania maximom pre vstup 2x2 je znázornený na obr. 2.15

V príklade na obr. 2.15 je vstupná vrstva o veľkosti 4x4 zmenšená procesom združovania maximom na veľkosť 2x2.

Výstup pooling vrstvy býva obyčajne ďalej redukovaný aktivačnou funkciou **ReLU** (Rectified Linear Unit), alebo inou obdobnou nelineárnou funkciou. Matematický výstup ReLU funkcie je veľmi jednoduchý:



Obr. 2.15: Príklad združovania maximom pre vstupnú oblasť 2x2

$$y = max(0, x)$$

Priebeh funkciou ReLU nastaví všetky záporné hodnoty na 0. Poradie ReLU oproti pooling vrstve je zameniteľné. Napriek tomu sa funkcia ReLU väčšinou využíva až po pooling vrstve, nakoľko pooling zmenšuje celkový počet neurónov vrstvy.

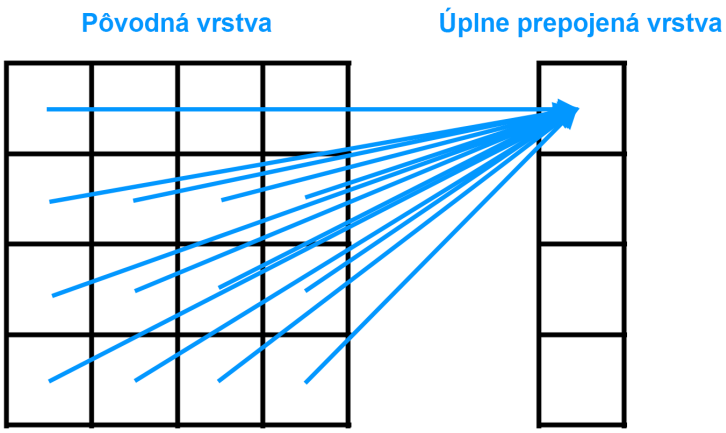
Úplne prepojená vrstva

Úplne prepojená vrstva (angl. fully-connected layer / FC) slúži na prepojenie výstupu z predošlých pooling alebo ReLU vrstiev. Vstupom sú teda aktivačné mapy týchto vrstiev a výstupom býva pravdepodobnostný vektor. Účelom úplne prepojnej vrstvy je spojiť všetky neuróny predošlej vrstvy s každým zo svojich neurónov, nakoľko všetky predošle konvolučné vrstvy fungujú len na princípe klasifikácie samostatných oddelených príznakov. Úplne prepojené vrstvy tým pádom väčšinou bývajú v architektúre konvolučnej siete umiestnené na konci. Príklad takejto vrstvy možno vidieť na obr. 2.16

Klasifikácia

Zatriedenie vstupu prebieha na záver vo výstupnej FC vrstve. Existuje viacero možných spôsobov, ako znormovať výstup klasifikácie do tried. Jedným z nich je funkcia *Softmax*. Ak konvolučná neurónová sieť vie klasifikovať presne K konkrétnych statických tried objektov (napr. mačka, pes, auto...), potom pre softmax funkciu platí nasledujúci vzťah, kde z je výstupný vektor z predposlednej vrstvy vstupujúci do Softmax vrstvy [26]:

$$S(z)_j = \frac{e^{-z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad pre\ j = 1, 2, ..., K \tag{2.7}$$



Obr. 2.16: Príklad úplne prepojenej vrstvy

Výstupom Softmax funkcie je pravdepodobnostný vektor, ktorý vyzerá napríklad takto: [0 .03 .01 0 0 .03 .93]. Výstupný pravdepodobnostný vektor značí pravdepodobnosť, že vstup patrí pod danú triedu pre každú z možných tried. Súčet všetkých pravdepodobností vo vektore musí byť rovný 1 (plných 100% je teda rozdelených do všetkých výstupných tried). Z vektoru v príklade je možné vyčítať niektoré znaky siete. Počet výstupných tried je 7. Pravdepodobnosť, vstup patrí do prvej triedy v poradí je nulová, pre druhú triedu je napríklad pravdepodobnosť 3%. Vieme povedať, že model v príklade zaraďuje vstup do poslednej triedy s vysokou mierou istoty. V podstate, výstupná úplne prepojená vrstva sa pozrie, ktoré zistené príznaky vstupu najviac korelujú s ktorými triedami a podľa toho zostaví výstupný pravdepodobnostný vektor.

Dropout vrstva

Princíp spočíva vo "vyradení" množiny neurónov procesom náhodného výberu. V skutočnosti nastaví hodnoty týchto neurónov na 0. Výsledkom je určitá redundancia pri pokuse o klasifikáciu vstupu. Sieť je donútená lepšie sa naučiť klasifikovať vstupy s nevýraznými príznakmi. Dropout je jeden zo spôsobov na zníženie overfittingu siete, spomínaného v podsekcii Pooling vrstva.

2.5 Nástroje pre tvorbu webových aplikácií

Cieľom práce je implementácia webovej aplikácie využívajúca model neurónovej siete pre zatriedenie audio skladby do žánru. V tejto sekcii sú opísané niektoré relevantné technológie pre tvorbu webových aplikácií na báze `JavaScript`-u alebo `Python` frameworkov.

JavaScript

JavaScript je vysoko-úrovňový dynamicky typovaný interpretovaný jazyk. Jadro jazyka JavaScript definuje minimálnu verziu API pre prácu so základnými typmi premenných a neobsahuje I/O operácie alebo iné sofistikovanejšie funkcie. Časť JavaScriptu-u pre prácu

s úložiskom, grafikou či sieťové záležitosti je definovaná prostredím, na ktorom JavaScript beží. Vo väčšine prípadov to býva internetový prehliadač. Základným účelom jazyku JavaScript je sprostredkovanie interakcie medzi užívateľom a webovou aplikáciou [15].

Základným účelom jazyku JavaScript je sprostredkovanie interakcie medzi užívateľom a webovou aplikáciu. JavaScript je jednovláknový programovací jazyk. Z tohto dôvodu je vo svojej prirodzenej forme synchrónny. Zaistenie asynchrónnosti pri niektorých procesoch je ale nevyhnutná časť z pohľadu UX (User Experience) dizajnu. Existencia asynchrónnosti je v JavaScripte prítomná od implementácie tzv. `callback`-ov. V druhej veľkej revízie JavaScriptu pod názvom ES6, alebo ECMAScript 2015, bola asynchrónnosť ďalej rozšírená pridaním triedy `Promise`. Od roku 2017 existuje kombinácia `async/await`.

Aj napriek tomu, že JavaScript sa rozrastá rýchlym tempom a stále pribúda nová funkcionalita, pre účely väčšiny webových aplikácii je samotný JavaScript v základnej forme nepoužiteľný. Tento problém je riešený existenciou rôznych knižníc a nástrojov pre ich správu. Jedným z takýchto nástrojov je `npm` (Node Package Manager). Primárne vytvorený pre `node.js`, open-source prostredie sprostredkujúce JavaScript na serveri (v backendu), npm ponúka prístup k veľkému množstvu užitočných JavaScript knižníc.

TensorFlow.js

TensorFlow[2] je open-source knižnica pôvodne vyvinutá spoločnosťou Google pre veľké numerické výpočty. Má hlboké možnosti využitia v oblasti umelej inteligencie a strojového učenia [8]. Zjednodušuje tvorbu neurónových sietí sprostredkovaním rozhrania medzi programátorom a samotnou sieťou. TensorFlow ponúka Python API pre vysoko-úrovňovú abstrakciu budovania modelov, pričom jednotlivé procesy a matematické operácie vykonáva v efektívnejšom C++. Od roku 2019 používa `Keras` API pre účely trénovania modelu [35].

Od vydania TensorFlow verzie 2.0 v roku 2019 bola pridaná mimo iné optimalizačné a abstrakčné zmeny aj knižnica pre JavaScript s názvom TensorFlow.js. Jej využitie je možné cez `npm`. Využíva hardvérovú akceleráciu pre budovanie modelov rovno v prehliadači. Napriek tomu, že samotné učenie modelu je cez túto knižnicu možné, jej hlavným využitím je nasadenie už naučeného modelu do webovej aplikácie pre použitie koncovým používateľom. Je teda možné vybudovať a naučiť model od základu v prostredí Python, uložiť dáta váh do súboru formátu HDP s príponou .h5 a následne konvertovať tento súbor do formátu JSON pre využitie v prostredí TensorFlow.js.

Django a Flask

`Django` a `Flask` sú najpoužívanejšie vysokoúrovňové nadstavby pre tvorbu a nasadenie rozšíriteľných webových aplikácií v prostredí jazyka Python. V niektorých stránkach sú si podobné a v iných majú rozdiely. Táto podsekcia spomenie niektoré z nich [32].

Django je full-stack robustný nástroj pre tvorbu veľkých a bezpečných webových aplikácií. Robustný z toho dôvodu, že takmer všetky nástroje potrebné pre tvorbu aplikácie ponúka už vo svojej základnej forme. Je to teda univerzálna nadstavba určená pre tvorbu aplikácií všetkých druhov. Dostupná je široká dokumentácia. Django automaticky riadi štandardné bezpečnostné procesy ako sú napr. správa účtov a transakcií. Je založený na využívaní návrhových vzorov pre tvorbu ľahko-udržiavateľného kódu. Nakoľko Django je

[2]TensorFlow sám o sebe pod nástroje webových aplikácií nepatrí. V originálnej podobe je to knižnica pre Python, Javu a C++. Nakoľko ale od roku 2019 existuje aj verzia pre JavaScript, bol zaradený do sekcie nástrojov pre webové aplikácie.

framework založený na komponentoch, každá vrstva je od seba oddelená, čo činí aplikácie vytvorené Djangom škálovateľnými na každej úrovni. Obsahuje systém ORM (Object Relational Mapper) pre uľahčenie prístupu a manipulácie s databázami.

Flask je ne druhú stranu výpočtovo-nenáročný, rozšíriteľný framework niekedy kategorizovaný ako micro-framework. Prvotná nenáročnosť použitia ho činí vhodnejším pre menšie aplikácie. Na rozdiel od Djanga, Flask je vo svojej základnej forme veľmi "holá" nadstavba, ktorú je možno rozšíriť veľkým počtom dostupných knižníc či pluginov. Flask teda ponúka väčšiu flexibilitu pri vývoji webových aplikácií. Rovnako ako je tomu pri Djangu, Flask obsahuje širokú dokumentáciu. Flask neobsahuje žiadny ORM systém, pre prácu s databázami je preto nutné využitie vonkajšieho nástroju ako je napr. `SQLAlchemy`.

Kapitola 3

Konštrukcia datasetu

Voľba vhodného datasetu je pri algoritmoch strojového učenia obzvlášť dôležitá časť. Aj napriek perfektne skonštruovaným vrstvám modelu neurónových sietí, ak je dataset nevhodný pre daný problém, výsledky nikdy nebudú zodpovedať pozitívnym očakávaniam. Táto skutočnosť platí aj pre model tejto práce.

3.1 Špecifikácia potrieb datasetu

Zadaním práce je vytvoriť model schopný rozpoznávať žáner skladby na základe nahrávky alebo audio súboru. Dataset bude tvorený kolekciou audio súborov rôznych skladieb z konečného počtu vopred definovaných žánrov veľkého (ideálne radovo tisíce) množstva. Pri nadmerných počtoch sa síce vyskytnú klesajúce výnosy ("diminishing returns"), kedy sa model z nových skladieb naučí exponenciálne menej než z prvých. V zásade ale platí – čím viac skladieb, tým lepší model.

Skladby by mali mať uspokojujúcu kvalitu. Pri nižšej kvalite nahrávok / audio súborov dochádza k stratám dôležitých znakov potrebných pre správne rozpoznanie žánru. Zároveň je dôležitá minimalizácia veľkosti súborov. Pri veľkom množstve skladieb s nadmernou veľkosťou dát je možné naraziť na nedostatok zdrojov pri trénovaní modelu. Minimalizácia veľkosti súborov by nemala nastať na úkor kvality nahrávok.

Pre podobu trénovacieho datasetu je nevyhnutné rozdelenie do žánrov. Aj napriek tomu, že väčšina populárnych skladieb sa dá veľmi jednoducho, až "lajdsky", rozdeliť do populárnych (main-stream) žánrov len na základe počutia, toto rozdelenie by malo byť dostatočne dôveryhodné. Slovo dobrého kamaráta Fera, ktorý prisahá, že skladba Highway to Hell patrí pod rock, ale nie je si 100% istý, lebo raz počul, že je to blues, vo väčšine prípadov nie je dostačujúce. Mierne, menej časté odchýlky a nedostatky v tomto oddelení sú ale prípustné. Prečo?

Prípustnosť odchýlok v zaradení do žánru

Táto skutočnosť vyplýva z toho, že niektoré skladby je jednoducho náročné až nemožné zatriediť do konkrétneho žánru. Jedna skladba môže mať často prvky viacerých žánrov. Napríklad kapela *Steely Dan* je známa svojou kombináciou rocku, jazzu a blues v ich skladbách [7]. Pri takej skladbe je potom komplikované povedať, že jej hlavným žánrom je jazz a tým pádom patrí pod jazz. Takýto postup by nebol úplne korektný.

Ďalšou príčinou nepresností môže byť jednoducho obšírnosť daného žánru. Pod hudobný žáner *rock* patrí nespočetne veľa skladieb a zaradiť všetky správne je prakticky nemožné.

Žánre sa postupom času vyvíjajú a vetvia. Vzniká množstvo progresívnych a indie žánrov, ktoré síce možno znejú ako rock, ale v skutočnosti je to progresívny indie punk hard-rock s prvkami jazzového metalu...

Nevyvrátiteľnou skutočnosťou, ktorá nadväzuje na predošlí odsek, je to, že žánre sa priebehom času menia. Skladbu, ktorú by sme 20 rokov naspäť nazvali popovou už nemusí nevyhnutne patriť pod pop teraz.

Potreby v bodoch

Pre účely práce by dataset mal vyzerať nasledovne:

- Veľké množstvo audio súborov skladieb rozdelených do žánrov (radovo tisíce),
- kvalita audio nahrávok s dostatočnou vzorkovacou frekvenciou a bitovou hĺbkou,
- čo najmenšia veľkosť dát,
- maximalizácia kvality a minimalizácia veľkosti súborov.

3.2 Dostupné riešenia

Dostupné riešenia datasetu pre danú úlohu sú pomerne problematická časť práce. Práca s veľkým množstvom skladieb je sama o sebe neľahká úloha. Každý súbor sa môže pohybovať od niekoľko MB až po 100 MB a viac pri vysokej kvalite audio súboru nekomprimovaného formátu. Ďalším neodvrátiteľným problémom je fakt, že dostupných datasetov je jednoducho málo. Otázka autorských práv pri distribúcii skladieb takýmto spôsobom nie je úplne jednoznačná a archívy hudby vo väčšine legitímnych prípadoch nie sú dostupné zadarmo. A ak takéto kolekcie dostupné sú, roztriedenie do žánrov alebo ich označenie je otázne.

Najpoužívanejším datasetom pre účely audio analýzy žánrov a prakticky jedinou „ready-to-use“ kolekciou je dataset **GTZAN**.

Dataset GTZAN

GTZAN je najpoužívanejším verejným datasetom na vyhodnotenie výskumu strojového počúvania na rozpoznávanie hudobných žánrov (MGR). Súbory boli zozbierané v rokoch 2000 – 2001 z rôznych zdrojov vrátane osobných CD, rádia, mikrofónových nahrávok – na reprezentáciu rôznych podmienok nahrávania [28].

Dataset pozostáva z 1 000 zvukových stôp, každá s dĺžkou 30 sekúnd. Obsahuje 10 žánrov, z ktorých každý je tvorený zo 100 skladieb. Všetky stopy sú 22050 Hz Mono 16-bitové zvukové súbory vo formáte .wav [27].

Obsahuje nasledujúce žánre: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock.

Dataset bol vytvorený *Georgom Tzanetakis*-om a *Perry R. Cook*-om pre účely ich výskumu *„Musical Genre Classification of Audio Signals“* [33].

MSD – Million Songs Dataset

MSD je voľne dostupná kolekcia audio znakov a metadát pre milión súčasných skladieb [10]. Celý dataset o veľkosti 300GB obsahuje dáta skladieb z prakticky každého časového obdobia a každého žánru, pre učenie základného modelu až zbytočne veľké množstvo. Dostupná je

ale aj podmnožina datasetu obsahujúca približne 10 000 skladieb (1% z celého datsetu), pre účely tejto práce viac než dostačujúca. Využitie tejto podmnožiny následne otvára možnosť rozšírenia modelu o zvyšné skladby datasetu.

Napriek tomu, že MSD v jeho základnej podobe neobsahuje rozdelenie do žánrov, existuje nad ním rozšírenie zvané *tagtraum*[30], ktoré túto úlohu spĺňa.

Vážnym nedostatkom tohto datasetu je fakt, že neobsahuje samotné nahrávky alebo audio súbory skladieb, ktoré opisuje. Je to dataset znakov ako napr. hlasitosť a metadát zozbieraných z data platforiem pre hudbu ako napr. *The Echo Nest*, *MusicBrainz*...

Jednou možnosťou je využitie dát dostupných v tomto datasete pre deep learning model. Pre model založený na CNN postupe už dostupné metadáta nie sú dostačujúce.

Keďže dataset obsahuje ID skladieb pre platformy, z ktorých bol vyzbieraný, zaujímavým riešením je využitie služieb platformy, ktorá funguje aj ako music provider. Takéto riešenie už bolo v minulosti preskúmané používateľom Matan Lachmish [21]. Platforma 7Digital neposkytuje svoje služby v oblasti Česka / Slovenska. Pre účely tejto práce je teda nutné využitie iného providera, prípadne iného riešenia datasetu.

Million Songs Dataset bol vydaný v spolupráci s The Echo Nest a používa Echo Nest identifikátory na označenie každej skladby. V júni 2016 vyplo Echo Nest svoje API a nezostala k dispozícii žiadna služba, ktorá by rozumela ID poskytnutým datasetom MSD.

Dôležitou službou poskytovanou prostredníctvom Echo Nest API bola služba Rosetta Stone, ktorá poskytovala mapovanie medzi Echo Nest ID a ID z iných hudobných služieb (pre účely práce sa jedná hlavne o Spotify).

Projekt AcousticBrainz skompiloval dataset dotazov na túto API pred jej ukončením [9].

spotDL

Pri preskúmavaní využitia služby *Spotify* ako riešenie music providera v kombinácii s datasetom MSD pre tvorbu konečného datasetu je ľahké naraziť na spotDL. Spotify Downloader je command-line aplikácia pre získavanie .mp3 súborov s metadátami (vrátane spotify žánru) zo spotify URL skladieb alebo playlistov/albumov. Funguje na princípe dotazovania Spotify API s ID skladby, pričom z odpovede zachová žiaduce metadáta a názov skladby. S názvom skladby následne dotazuje YouTube vyhľadávač a z najlepšieho výsledku uloží zvukovú stopu vo formáte .mp3.

Je teda možné vytvoriť dataset len na základe tejto aplikácie a následného využitia žánru v metadátach pre vytvorenie trénovacej množiny.

Problém nastáva pri výbere skladieb, ktoré budú tvoriť konečný dataset. Takýto spôsob tvorby datasetu by bol vhodný pri tvorbe modelu pre identifikáciu subžánrov niektorého konkrétneho žánru, resp. tvorbe modelu, ktorý by sa mal učiť z konkrétnych vybraných množín (playlistov/albumov), napr. model ktorý by predikoval obľúbenosť skladby na základe preferencií používateľa.

FMA

Free Music Archive (FMA) [11], je otvorený a ľahko dostupný komplexný dataset vhodný na vyhodnocovanie niekoľkých úloh v MIR (Music Information Retrieval), oblasti týkajúcej sa prehliadania, vyhľadávania a organizovania veľkých hudobných zbierok. Rieši problém nízkej dostupnosti datasetov so zvukovými stopami. V plnej veľkosti poskytuje až 917 GB zvukových stôp v počte 106 574 skladieb od 16 341 umelcov zo 161 žánrov usporiadaných v hierarchickej forme. Poskytuje zvuk v plnej dĺžke a vysokej kvality, vopred vypočítané

funkcie spolu s metadátami na úrovni stôp a používateľov, značkami a textom voľnej formy, ako sú biografie. Keďže bol dataset vytvorený pre úlohy algoritmov učenia, obsahuje odporúčané rozdelenie na trénovacie a testovacie množiny.

Existujú aj menšie, dostupnejšie množiny datasetu, nakoľko celý dataset má až 917 GB.

- fma_small – 8000 skladieb o dĺžke 30s, 8 vyvážených žánrov - 7,2 GB
- fma_medium – 25000 skladieb o dĺžke 30s, 16 nevyvážených žánrov – 22 GB
- fma_large – 106574 skladieb o dĺžke 30s, 161 nevyvážených žánrov - 93 GB
- fma_full – 106574 skladieb plnej dĺžky, 161 nevyvážených žánrov - 879 GB

Problémy dostupných riešení

Zásadným problémom pri riešení tejto časti úlohy je nedostatok dostupných datasetov. Je to problém, od ktorého sa v základe odvíjajú problémy ostatné. Nájsť dataset, ktorý úplne spĺňa všetky dané požiadavky je neľahká záležitosť. Častým háčikom býva príliš malé množstvo dostupných skladieb či žánrov, čo by mohlo spôsobiť buď overfitting siete ak sú skladby príliš podobné, alebo prípadne nedostatočné naučenie modelu celkovo. Pri postupe založenom na .wav súboroch sa tento problém násobí. Najväčším problémom je nízka dostupnosť datasetov obsahujúcich samotné audio súbory v kombinácii s zaradením do žánrov. V takomto prípade existuje možnosť kombinácie viacerých datasetov do jedného, ale tento spôsob je pomerne nespoľahlivý a nerieši problém v jeho základe.

MP3, WAV či iné?

Pri výbere vhodného datasetu pre prístup založený na CNN je dôležité zobrať formát súborov do úvahy. Datasety obsahujúce .wav súbory disponujú kvalitnejšími nestratovými nahrávkami na úkor veľkosti súborov. Ak platí predpoklad, že kvalita mp3 súborov / iného stratového formátu je dostačujúca pre účely CNN modelu, využitie tohto formátu je preferované.

3.3 Zvolené riešenie

Dataset FMA najlepšie spĺňa požiadavky práce, a preto bol zvolený ako vyhovujúce riešenie. Obsahuje viac než dostatočné množstvo skladieb z viac než priaznivého počtu žánrov. Naviac zahŕňa uspokojujúcu dokumentáciu a príklady použitia. Existencia viacerých podmnožín datasetu tiež značí možnosti na rozšírenie práce, nakoľko využitie plnej veľkosti datasetu je pre naše účely neprimerané.

Obsahuje skladby formátu MP3 rôznej kvality. Táto skutočnosť je vyhovujúca, nakoľko aj audio súbory nahrané používateľmi na rozpoznanie vo finálnej podobe projektu môžu mať rôzny bitrate. Model by mal byť teda naučený na stopách aj rozdielnej kvality.

Kapitola 4

Príprava datasetu a preprocessing

Táto kapitola sa venuje krokom, ktoré je nutné urobiť a nástrojom, ktoré toto umožnia, na prípravu zvoleného datasetu do formy vhodnej pre trénovanie neurónovej siete. Na začiatok je žiaduce poznamenať, prečo sú tieto kroky potrebné pre splnenie úlohy. V podstate ide o to, že dostupné datasety bývajú buď pripravené na prácu z viacerých oblastí, alebo nebývajú v niektorých prípadoch nachystané vôbec. Pre využitie týchto dát na učenie neurónovej siete dáta často treba určitým spôsobom upraviť, zmeniť alebo vytriediť.

4.1 Príprava FMA datasetu

Dataset FMA je vhodný pre mnoho druhov úloh audio analýzy. Okrem rozdelenia do žánrov obsahuje aj veľké množstvo príznakov a metadát, ktoré sú pre účely tejto práce nepodstatné. Prvým krokom je "vyčistenie" datasetu od nepotrebných údajov a vytriedenie audio skladieb do zložiek podľa žánru. Python modul `numpy` v kombinácii s `pandas` sú dôležitým nástrojom pre analýzu dát, pričom dataset FMA nie je výnimkou. Existuje voľne dostupný modul `utils.py`, ktorý je súčasťou FMA datasetu a jeho účelom je uľahčenie a zefektívnenie práce s datasetom. Vytvára rozhranie medzi programátorom a príznakmi, dátami či audio súbormi obsiahnutými v datasete. Všetka práca s dátami datasetu prebieha s využitím týchto modulov.

Iteráciou cez všetky skladby (zvolenej podmnožiny datasetu, napr. small, medium a pod.) evidované v súbore `tracks.csv` je možné rozdelenie do zložiek podľa žánrov. Takýmto spôsobom vzniká niečo ako medzidataset, ktorý je pripravený na ďalšiu prácu, menovite preprocessing a následné učenie modelu.

Nakoľko dataset FMA obsahuje veľmi veľké množstvo skladieb a žánrov, využitie celej veľkosti datasetu je pre túto prácu nadbytočné. Z tohto dôvodu je pre vytvorenie modelu schopného klasifikovať audio nahrávky hudby do najpopulárnejších žánrov vhodné zvoliť dataset s označením `small`, teda podmnožinu malej veľkosti. Táto podmnožina obsahuje celkovo 8000 skladieb dĺžky 30 sekúnd rozdelených do 8-ich žánrov. Pre vytvorenie modelu schopného klasifikovať podžánre *elektronickej* hudby je nutné využitie datasetu väčšej podmnožiny, menovite `large`, ktorá obsahuje všetky žánre a skladby dostupné v datasete, s jediným obmedzením orezania skladieb na dĺžku 30 sekúnd na rozdiel od celej dĺžky skladieb v plnej veľkosti datasetu.

V tabuľke. 4.1 je znázornený obsah súboru `genres.csv`, ktorý spisuje anotácie ku všetkým žánrom obsiahnutým v datasete. Hlavičky stĺpcov boli preložené a stĺpec s farebným označením vynechaný (pre túto prácu je nepodstatný).

Tabuľka 4.1: Znázornenie obsahu súboru genres.csv.

ID žánru	Označenie žánru	ID nadradeného žánru	Názov žánru
1	Avant-Garde	38	Avant-Garde
2	International		International
3	Blues		Blues
4	Jazz		Jazz
5	Classical		Classical
...			
15	Electronic		Electronic
...			
181	Techno	15	Techno
182	House	15	House
...			

Ako je v tabuľke možné vidieť, dataset obsahuje pre každý žáner, mimo žánrov prvej úrovne, ID nadradeného žánru. Príprava datasetu pre učenie neurónovej siete schopnej klasifikovať podžánre elektronickej hudby by potom prebiehala nasledovne:

1. Zvolená podmnožina datasetu musí byť nastavená na `large`.
2. Iterovaním sa prechádzajú všetky skladby obsiahnuté v súbore `tracks.csv`
3. Ak ID žánru nadradeného žánru danej skladby je zhodné s ID žánru elektronickej hudby, teda 15, potom je nutné audio súbor skladby premiestniť do novej zložky medzidatasetu.
4. Ak zložka daného žánru ešte neexistuje, potom je nutné ju vytvoriť.
5. Súbor skladby je presunutý do podzložky s názvom žánru, do ktorého patrí.
6. Návrat naspäť na krok 3., až dokým neboli iterovaním prejdené všetky skladby.

Po vykonaní týchto krokov a podobného, no menej zložitého, procesu pre podmnožinu datasetu veľkosti `small` spomínaného vyššie sú dataset elektronickej hudby a dataset 16-ich populárnych žánrov pripravené na preprocessing.

4.2 Preprocessing

Zaradenie dvoch skladieb do rovnakého žánru v zásade prebieha na základe nejakých určitých rozoznateľných podobností. Napríklad kombináciu buildup-u (časť skladby, kedy hudba postupne naberá na sile alebo tempu, resp. buduje k niečomu) a následného dropu (najsilnejšia časť skladby) v elektronickej hudbe je možné rozoznať podľa postupného pomalého vzostupu amplitúdy, krátkeho poklesu až odmlku a následného vyskočenia amplitúdy na najvyššie hodnoty. Spôsobilosť konvolučných neurónových sietí na rozpoznávanie takýchto opakujúcich sa vzorov je pri úlohe klasifikácie do žánrov veľmi žiadúca vlastnosť.

Faktom ale je, že konvolučné siete zásadne prijímajú vstup v obrázkovej forme. Ďalším krokom v príprave dát je prevedenie skladieb z audio formy do formy obrázkovej. Jednou z možností je grafovanie priebehu signálu. Pri použití tohto spôsobu nesie graf informáciu

o amplitúde za čas. Mnohonásobne lepším spôsobom je ale využitie spektrálnej charakteristiky zvuku pre vytvorenie spektrogramov. Spektrogram, na rozdiel od grafu priebehu signálu, obsahuje naviac informáciu o frekvencii signálu.

Pre transformáciu audio súborov na súbory obrazové je možné využiť modul `librosa` v kombinácii s modulom `Matplotlib`. Librosa [23] je python knižnica určená pre mnoho druhov audio analýzy. Obsahuje efektívne nástroje pre načítanie, spracovanie a úpravu audia rôznych veľkostí a kvality. Jej funkcia pre načítanie audio súboru vráti `Pandas` radu typu float32 a informáciu o vzorkovacej frekvencii audia. Znalosť vzorkovacej frekvencie je dôležitá pre nasledujúce kroky preprocessingu. Séria je ďalej rozdelená na intervaly zodpovedajúce dĺžke 5 sekúnd. V tomto momente je vhodné poznamenať dôvod rozdelenia audia na 5-sekundové intervaly. V krátkosti ide o dva dôvody. Prvým je lepšie rozlíšenie výsledného spektrogramu. Je logické, že čím viac pixelov je pre každý spektrogram použitých, tým väčší je počet neurónov potrebných na extrakciu príznakov a tým náročnejší je proces učenia konvolučnej siete na výpočtové zdroje. Cieľom je teda nájsť vhodné rozlíšenie spektrogramov pre minimalizáciu veľkosti a maximalizáciu informácie, ktorú nesú. Spektrogramy s menším rozlíšením dokážu niesť väčšie množstvo informácií pre audio s dĺžkou 5 sekúnd v porovnaní s audiom, ktoré má dĺžku 90 a viac sekúnd (pre dĺžku celej skladby) alebo 30 sekúnd (pre prípad podmnožiny `large` FMA datasetu). Ďalším dôvodom je, že takto naučený model bude lepšie pripravený na rozpoznanie žánru aj krátkeho úseku skladby.

Pri klasifikovaní skladby dĺžky viac než 5 sekúnd je audio rozdelené na čo najväčší počet 5-sekundových intervalov, pričom zvyšok o dĺžky $< 5s$ je zahodený. Všetky intervaly sú ohodnotené zvlášť a pre výsledné určenie žánru je prevedený priemer všetkých zložiek všetkých pravdepodobnostných vektorov.

Pre tvorbu samotných spektrogramov je nutné využitie jedného z typov Fourierovej transformácie, menovite **STFT**. Librosa ponúka funkciu aj pre tento účel, prevedenie série na spektrogram je teda pomerne priamočiare. Ďalším krokom je prevedenie amplitúdovej stupnice na stupnicu decibelov. Týmto krokom vznikne niečo ako mel spectrogram, pričom librosa používa iné konštanty na prepočet decibelov. Zásadný účel ale zostáva rovnaký. Kde klasické STFT používa lineárnu stupnicu frekvencií, spektrogram vytvorený konverziou amplitúdy na decibely librosa funkciou má decibelovú stupnicu logaritmickú. Prevedením na logaritmickú stupnicu je lepšie využitý priestor spektrogramu pre dôležitejšie príznaky skladby, ako bolo spomenuté v sekcii Mel spektrogramy.

Spektrogram je následne možné premietnuť do grafu pomocou modulu Matplotlib[1]. Pre učenie konvolučnej siete je podstatné len samotné vnútro spektrogramu. Anotácie osí a stupnice sú z tohto dôvodu zahodené. Graf spektrogramu je potom uložený vo formáte PNG do novej zložky so spektrogramami. Tento proces je vykonaný pre všetky skladby získané z datasetu FMA. Dôvod oddelenia procesu tvorby spektrogramu z audia a samotného učenia siete je možnosť spustenia celého preprocessingu oddelene pomocou argumentov pri spúšťaní výsledného python modulu. Pri klasifikácii vstupu naučeným modelom vo výslednej aplikácii prebehne preprocessing podobnou, analogickou formou pre jednu skladbu.

[1] Je dôležité spomenúť, že na niektorých systémoch, menovite Mac OS X (na ktorom bola táto práca vytvorená) a po pripojení cez službu ssh, modul Matplotlib má známu chybu úniku pamäte pri vytváraní veľkého množstva grafov. Tento únik je možné opraviť zmenou využitého backendu pomocou príkazu *matplotlib.use('TkAgg')*, kde "TkAgg" je zameniteľné s inými backendmi, napr. *Agg*, *Cairo* a podobne.

Kapitola 5

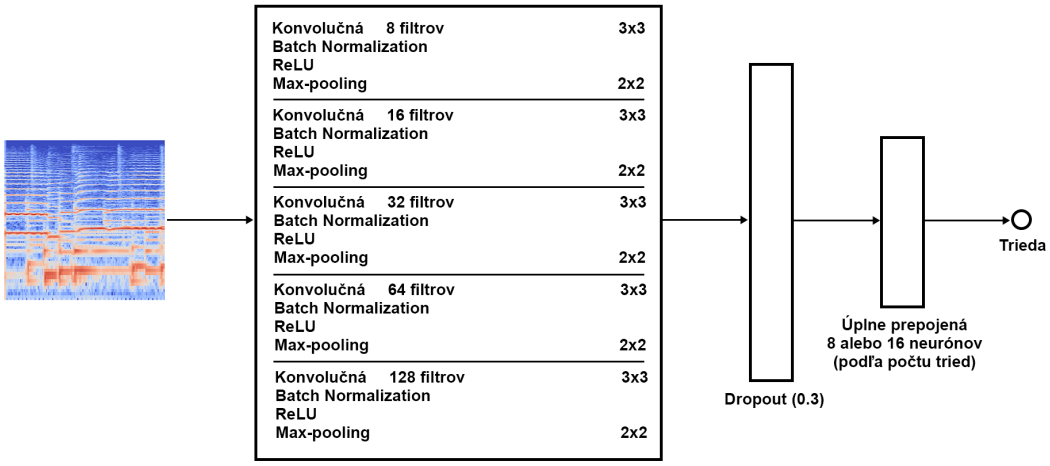
Návrh siete a učenie

V rámci práce boli vytvorené dva modely konvolučnej neurónovej siete predikujúce žáner skladieb. Prvým bol model všeobecný, pre všeobecne populárne žánre a druhým bol model sústreďujúci sa na podžánre elektronickej hudby. Aj napriek tomu, že je možné vytvoriť jeden model, ktorý bude učený ako na všeobecných žánroch tak aj na podžánroch jedného alebo viacerých z nich, vytvorenie individuálneho modelu pre podkategórie konkrétneho žánru by teoreticky malo preukázať lepšie výsledky. Tento myšlienkový pochod vyplýva z faktu, že rozdiely medzi jednotlivými všeobecnými žánrami sú podstatne väčšie než rozdiely v rámci podžánrov jedného individuálneho žánru. Pre obidva naučené modely bola využitá rovnaká architektúra konvolučnej siete opísaná v nasledujúcej sekcii. Okrem návrhu siete sa v tejto kapitole nachádza informácia o učení jednotlivých modelov.

5.1 Návrh siete

Navrhnutá architektúra siete je v základe zložená z niekoľkých konvolučných vrstiev pre identifikáciu príznakov a jednej úplne prepojenej výstupnej vrstvy na konci. Medzi jednotlivými konvolučnými vrstvami, resp. po každej z nich, sú zaradené ReLU vrstvy spolu s Max-Pooling vrstvami s účelom podvzorkovania vstupu. Po každej konvolučnej vrstve je tiež využitá dávková normalizačná (angl. *Batch Normalization*) vrstva pre aplikáciu transformácie, ktorej cieľom je udržať strednú hodnotu výstupu pri nule a výstupnú štandardnú odchýlku blízko jednej. V zásade ide o normalizáciu prípadných rozptýlených alebo jednostranných hodnôt smerom k nule. Jej zaradenie do architektúry siete čo sa týka pozície je prinajmenšom sporné. Niektoré zdroje [18] uvádzajú, že normalizácia by mala prebehnúť pred akoukoľvek nelineárnosťou, iné [24] zase, že jej využitie po aktivácii ReLU funkcie zvyšuje presnosť siete a znižuje stratu. V modeli tejto práce je normalizácia aplikovaná pred aktiváciou ReLU. Prítomnosťou normalizačných vrstviev je možné vypustiť Dropout vrstvy po každej vrstve konvolučnej, nakoľko zavedenie normalizácie má podobné regulačné účinky na overfitting siete ako Dropout vrstva [22]. Vypustením Dropout vrstvy je zároveň docielené zvýšenie rýchlosti učenia siete. Prítomná je jediná dropout vrstva pred výstupnou úplne prepojenou vrstvou. U výstupnej vrstvy je využitá aktivačná funkcia softmax. Výstupné hodnoty funkcie softmax ležia medzi 0 a 1 a udávajú mieru, v akej vstup patrí do každej z tried.

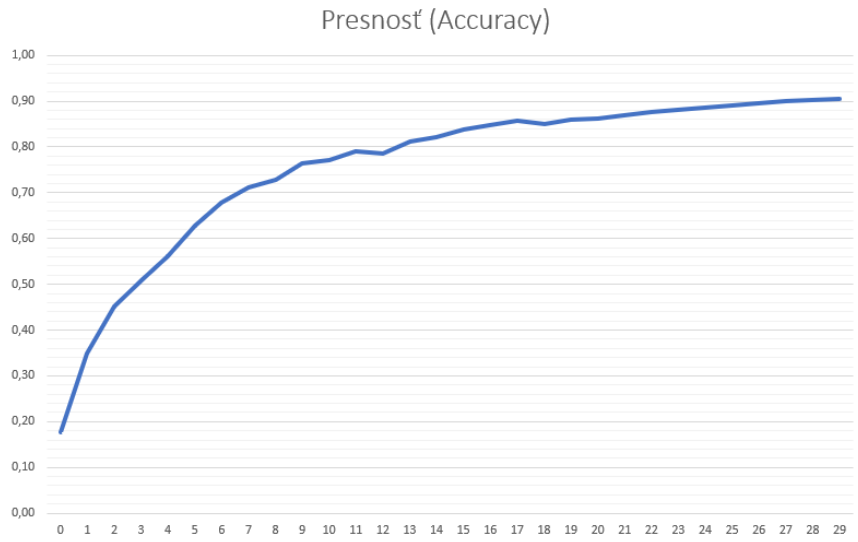
Pomer rozdelenia vstupných dát na trénovaciu a testovaciu množinu bol nastavený na 80%/20%.



Obr. 5.1: Znázornenie architektúry siete

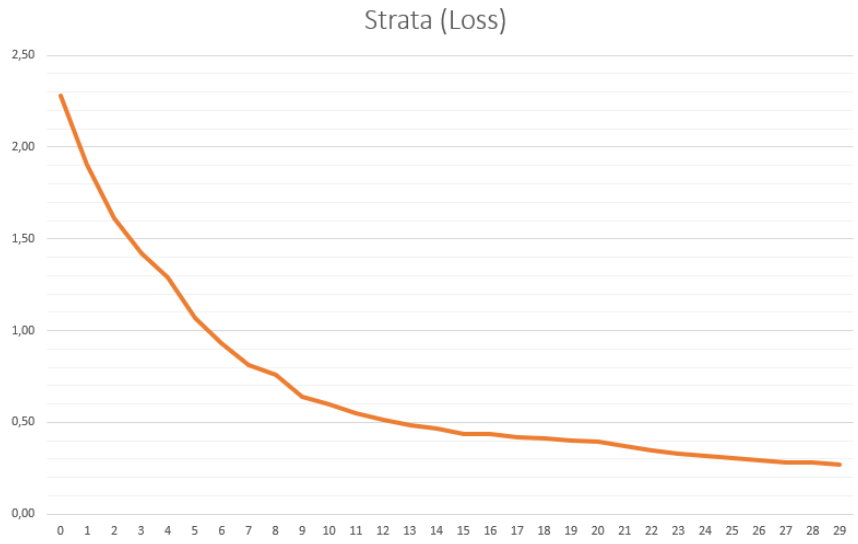
5.2 Učenie všeobecnej siete

Pre učenie modelu všeobecnej siete bol využitý dataset `FMA` podmnožiny `small`, obsahujúci celkom 8 vyrovnaných tried. Bližšie informácie ku konkrétnej podmnožine datasetu sú opísané v sekcii 3.2 Dostupné riešenia kapitoly 3 Konštrukcia datasetu.



Obr. 5.2: Graf presnosti počas priebehu učenia všeobecného modelu

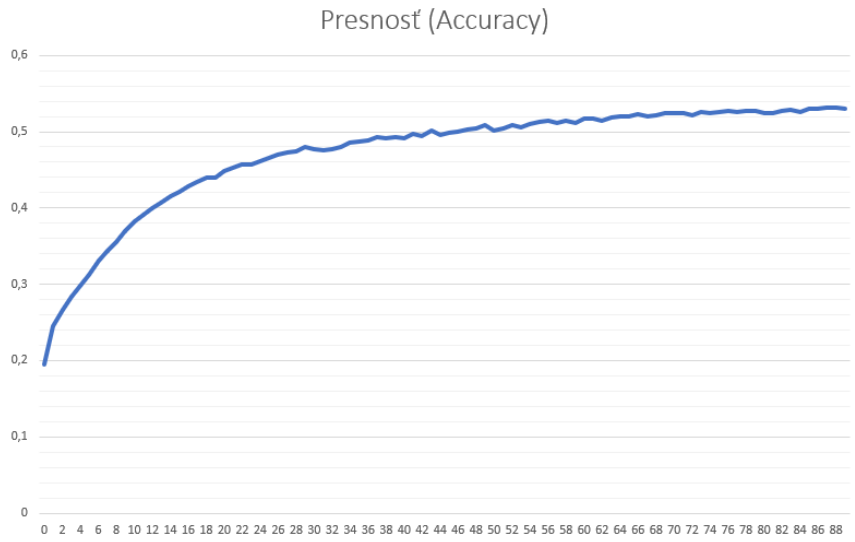
Ako je možné vidieť na obrázkoch 5.2 a 5.3, model na konci učenia dosiahol presnosti vyššej než $0,9$ a strata klesla pod hodnotu $0,3$. Celé učenie prebehlo v 30 epochách.



Obr. 5.3: Graf straty počas priebehu učenia všeobecného modelu

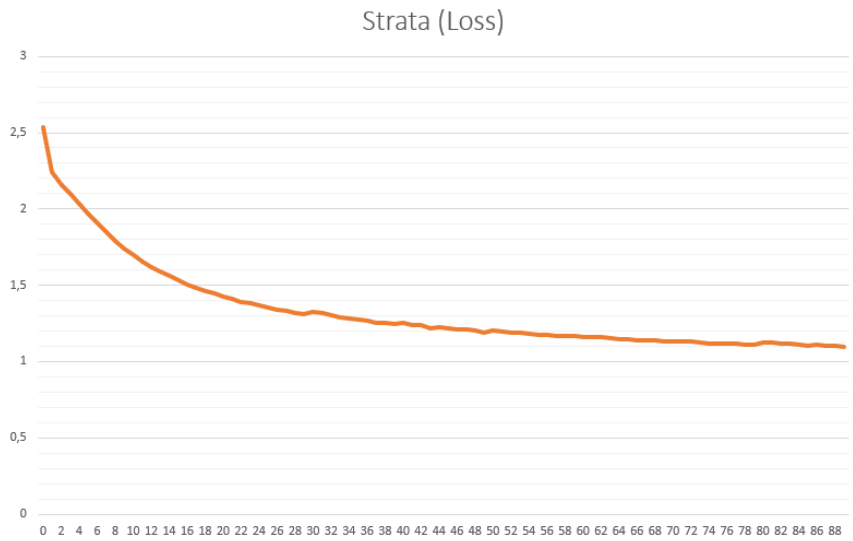
5.3 Učenie siete podžánrov elektronickej hudby

Model pre rozpoznávanie podžánrov elektronickej hudby bol taktiež učený na dátach z datasetu `FMA`, tentokrát sa ale jednalo o podmnožinu `large`, z ktorej boli vybrané všetky skladby patriace do niektorého z podžánrov elektronickej hudby. Celý proces výberu týchto skladieb je opísaný v sekcii 4.1 Príprava FMA datasetu patriacej do 4. kapitoly tejto práce.



Obr. 5.4: Graf presnosti počas priebehu čenia modelu podžánrov elektronickej hudby

Priebeh učenia modelu podžánrov elektronickej hudby je znázornený na grafoch presnosti na obr. 5.4 a grafu straty na obr. 5.5. Ako je možné na týchto grafoch vidieť, tento model dosiahol porovnateľne horších výsledkov než model všeobecný. Po 90 epochách uče-



Obr. 5.5: Graf straty počas priebehu učenia modelu podžánrov elektronickej hudby

nia, model dosiahol presnosti menej než $0,55$ (55%). Je možné predpokladať, že aj po veľkom množstve nasledujúcich epoch by model nedosiahol pomerne lepších výsledkov.

5.4 Vyhodnotenie

Horšie výsledky podžánrového modelu môžu byť zapríčinené viacerými faktami, z ktorých väčšina súvisí so vstupnými dátami datasetu `FMA`. Prvým dôležitým faktom je nevyrovnanosť jednotlivých žánrov elektronickej hudby dostupných v tomto datasete. Jednotlivé zložky spektrogramov po ich vytvorení naberali hodnôt od 150MB v minimálnom prípade po hodnoty 1,5GB v maximálnom prípade. Jedná sa teda o pomer 1:10 v najhorších prípadoch, čo je pomer zvládnuteľný využitím váh pri učení, ale napriek tomu vplývajúci na výslednú presnosť modelu.

Ďalším dôležitým faktom je zložitosť diferencovania podžánrov elektronickej hudby. Nakoľko všetky skladby majú rovnaký rodičovský žáner, skladby budú mať prinajmenšom podobné prvky. Odlíšiť tieto prvky medzi jednotlivými žánrami môže byť pre neurónovú sieť zložitejšie než odlíšiť samotné rodičovské žánre. Jednoduchšie povedané, rozdiely medzi rockovou skladbou a klasickou hudbou sú poznateľnejšie než rozdiely medzi podžánrami elektronickej hudby. Taktiež je možné, až veľmi pravdepodobné, že skladby v rámci elektronickej hudby čerpajú z viacero z jej podžánrov. Tento argument je podporovaný faktom, že skladby boli v datasete vo veľkom pomere zaradené pod viacero podžánrov. Toto mohlo zapríčiniť, že vytvorené spektrogramy nemali dostatočne veľké rozlíšenie pre uspokojujúce rozpoznanie rozdielov v jednotlivých žánroch.

Kapitola 6

Tvorba webovej aplikácie

Pre tvorbu webovej aplikácie na nasadenie konvolučnej siete pre použitie je využitý hlavne jazyk JavaScript v kombinácii s klasickým HTML a `tailwindcss`. Využitie jednej z Python nadstavieb pre tvorbu webových aplikácií bolo zvážené, konkrétne sa jedná o frameworky Django a Flask. Django je nástroj pre tvorbu robustných aplikácií s veľkými databázami, pre túto prácu je teda nevhodný. Čo sa týka nadstavby Flask, jej využitie už je dostupnejšie aj pre menšie aplikácie, nakoľko ale existuje knižnica TensorFlow.js, JavaScript pre účely práce bohate stačí.

6.1 Návrh aplikácie

Cieľom zadania je vytvoriť aplikáciu, ktorá bude schopná rozpoznávať žánre populárnych skladieb a zároveň prehrávať danú skladbu. Prvým krokom je teda definícia spôsobu, akým budú dané skladby vyberané na klasifikáciu používateľom. Existuje možnosť využitia API jedného z internetových poskytovateľov hudby (napr. Spotify), pre vytvorenie playlistu, či vyhľadávača playlistov, z ktorého si používateľ môže vybrať. Toto je vhodné skôr ako implementovateľné rozšírenie v budúcnosti. Oveľa prostejším riešením je ponúknuť používateľovi nahrať akúkoľvek skladbu pomocou HTML5 políčka `input`, ktorá následne bude klasifikovaná.

Po nahraní skladby je z input políčka vytiahnutá URL dočasného súboru JavaScriptom. Využitím JavaScriptu na strane klienta je možné sa vyhnúť nepotrebnému obnovovaniu stránky po odoslaní formuláru a manuálnemu ukladaniu súboru jazykom `php` na strane serveru.

Následne sa zobrazí panel s ovládaním pre prehrávanie skladby. Panel je vytvorený s použitím knižnice `green-audio-player`. Na obr. 6.1 je znázornená základná podoba tohto panelu.

Obr. 6.1: Znázornenie základnej podoby panelu z knižnice *green-audio-player*

Ďalším krokom je zaistenie priebehu preprocessingu na nahranej skladbe. Existuje možnosť využitia JavaScript-ových knižníc analogických k tým použitým v jazyku Python. Pre Python modul `librosa` by to bola knižnica `meyda`. Fungujúc na podobnom princípe

ako librosa, meyda je knižnica pre extrakciu audio príznakov formou buď offline režimu v prehliadači alebo pomocou dostupnej web API. Pre nahradenie modulu `Matplotlib` existuje prehliadačová Javascript verzia kompatibilná so zmieneným Matplotlib-om s názvom `mpld3`. Aj napriek tomu, že využitím týchto dvoch knižníc je možné dosiahnuť podobného výsledku ako pri tvorbe spektrogramov v prostredí Python, tak je veľká pravdepodobnosť že jednotlivé spektrogramy nebudú mať úplne rovnaký formát. Pre model naučený na spektrogramoch v prostredí Python je žiaduce aby aj vstupné spektrogramy klasifikovaných skladieb mali rovnaký formát. Z tohto dôvodu je preprocessing prevedený spustením Python skriptu. Spustenie tohto skriptu je riešené vzdialeným serverom v API formáte.

API

Vzdialená web API je hostovaná na serveroch AWS (Amazon Web Services) a pristupuje sa k nej pomocou JavasScript-ovej XHR (XMLHttpRequest) žiadosti. Verejná IP adresa serveru je `18.156.120.67`. Vzdialený server je obsluhovaný Python nadstavbou Flask. Tu je vhodné upozorniť na to, že Flask síce má vlastný vývojový server vytvorený ako nadstavba nad WSGI (Web Server Gateway Interface) pomocnou kolekciou knižníc `Werkzeug`, tento server ale nie je vhodný pre produkčné prostredie, nakoľko je obmedzený počtom žiadostí, ktoré dokáže spracovať. Pre väčšie množstvo žiadostí je teda neprispôsobený a v obyčajnom prípade je vhodné využiť jednu so serverových služieb pripravených pre produkčné prostredie, ako je napr. Python modul `waitress`. Keďže v prípade tejto práce nie je nutné sa obávať vysokého množstva žiadostí, server obsluhovaný nadstavbou Flask je dostačujúci.

Po prijatí dotazu správneho formátu s audio súborom dostupným pod parametrom s názvom **file**, API tento súbor uloží. Následne sa pokračuje podobným postupom ako pri príprave dát pre trénovanie konvolučnej siete. Audio súbor je znova načítaný knižnicou `librosa` do jedného pola, ktoré je následne rozdelené na 5-sekundové časti využitím vzorkovacej frekvencie audia, ktorú vracia načítavacia funkcia z knižnice librosa. Posledný zvyšok pola o dĺžke menej než 5 sekúnd je zahodený. Pre každú z týchto častí je vytvorený spektrogram a klasifikácia tiež prebehne individuálne. Pre klasifikáciu je využitý súbor formátu `HDF` (prípona .h5), ktorý obsahuje nastavené váhy po trénovaní konvolučnej siete. Po klasifikovaní každej z 5-sekundových častí je spravený priemer všetkých výsledných pravdepodobností.

Vo výsledku je teda vytvorená vzdialená API ktorá prijíma POST žiadosti na porte `5000`, na ktorom beží Flask aplikácia. Obsahom takejto žiadosti je samotný audio súbor pre klasifikáciu pod parametrom s názvom `file`. Žiadosť musí mať hlavičku, ktorá nastavuje formát obsahu žiadosti na `multipart/form-data`. API vracia odpoveď vo formáte `JSON` objektu s informáciou o pravdepodobnostiach pre jednotlivé triedy.

Na obr. 6.2 je znázornený príklad odpovede web API na dotaz. Je dôležité poznamenať, že API vracia len výsledky s percentuálnou hodnotou vyššou než **3**.

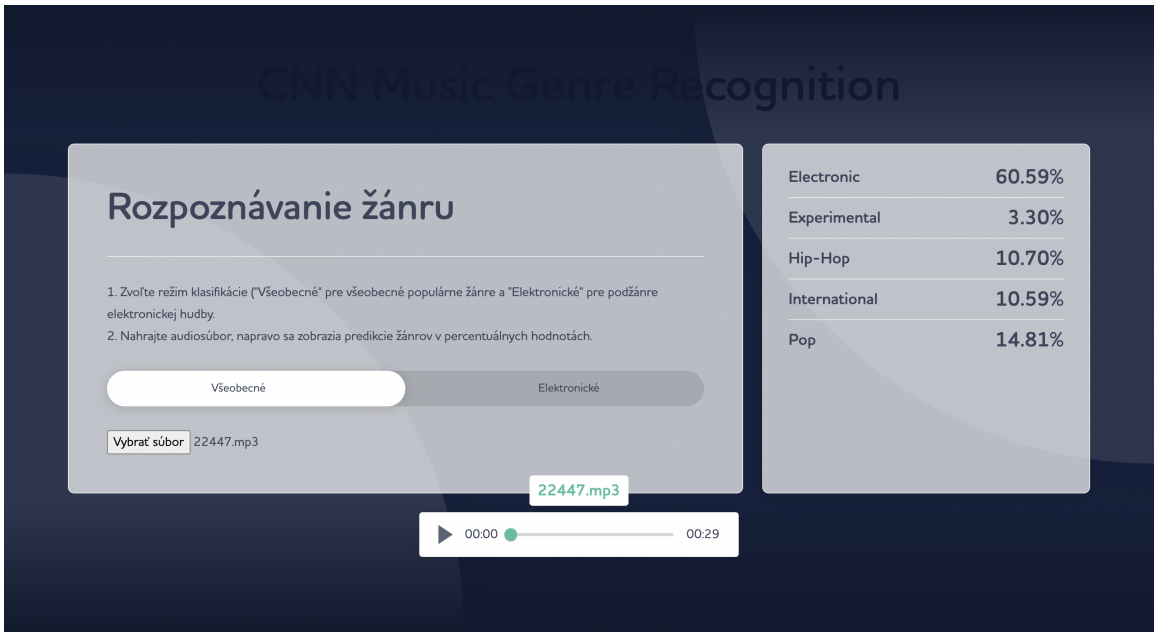
6.2 Výsledná aplikácia

Podoba výslednej aplikácie je zobrazená na obr. 6.3. Existuje možnosť prepnutia medzi "Všeobecným" a "Elektronickým" módom. Jednotlivé módy určujú, ktorá konvolučná sieť bude využitá pri predikcii žánru dát. Podľa stanoveného módu sa do POST žiadosti pridá parameter `mode` s hodnotou `general` alebo `electronic` a Python skript bežiaci na API zaobstará načítanie vhodného modelu.

```
{
    "result":
    {
        "Electronic":0.5524438619613647,
        "Experimental":0.03348640352487564,
        "Hip-Hop":0.259492963552475,
        "Pop":0.0702413022518158,
        "Rock":0.08111841231584549
    }
}
```


Obr. 6.2: Príklad JSON odpovede z API

Po získaní odpovede na XHR žiadosť vo forme znázornenej na obr. 6.2, sa jednotlivé predikované žánre zobrazia v pravej lište v aplikácii. Používateľ má medzitým možnosť prehrávať nahranú skladbu v dolnej časti aplikácie, pričom počas predikcie, ktorá trvá 5 – 10s, sa v pravej lište zobrazí načítavacia ikona. Celý proces prebieha bez akéhokoľvek obnovovania stránky v prehliadači.



Obr. 6.3: Podoba výslednej webovej aplikácie

Kapitola 7

Záver

Cieľom zadania bolo vytvorenie webovej aplikácie schopnej rozpoznávať žánre populárnych skladieb.

V rámci práce bol spísaný prieskum dostupných datasetov pre úlohy získavania informácií z hudby, vrátane rozboru ich prijateľnosti pre účel rozpoznávania žánru. Z dostupných datasetov bol zvolený dataset FMA ako najvhodnejší pre zadanú úlohu.

Pre účely splnenia zadania práce boli naštudované a spísané informácie o zvuku, jeho digitálnej charakteristike a strojovom učení so zameraním na konvolučné neurónové siete.

V rámci práce boli vytvorené dva modely konvolučnej neurónovej siete, jeden pre rozpoznávanie žánrov všeobecne populárnych a druhý pre podžánre elektronickej hudby. Model všeobecných žánrov dosiahol značne lepších výsledkov než model elektronickej hudby. Toto mohlo byť zapríčinené nevyrovnaným počtom skladieb v datasete či nevýraznými rozdielmi medzi jednotlivými podžánrami elektronickej hudby.

Pre tvorbu vizuálnych dát pre učenie konvolučnej siete zo samotných audio súborov skladieb datasetu bol definovaný proces tvorby spektrogramov vrátane knižníc využitých pre tento účel.

Konečným výsledkom je webová aplikácia, ktorá po nahraní audio súboru používateľom zobrazí pravdepodobnosti pre jednotlivé žánre všeobecne populárnych žánrov či žánrov elektronickej hudby podľa výberu používateľa.

Prácu je možné rozšíriť najmä využitím širšej podmnožiny datasetu FMA. Nakoľko pre všeobecný model bola využitá najmenšia dostupná podmnožina datasetu, model je možné rozšíriť učením na podmnožine strednej, veľkej alebo prípadne na podmnožine plnej veľkosti datasetu. Taktiež je možné vytvorenie ďalších podžánrových modelov pre iné žánre analogickým spôsobom ku spôsobu opísanom v tejto práci pre model podžánrov elektronickej hudby.

Literatúra

[1] *ANSAMBL* [online]. [cit. 2021-12-4]. Dostupné z: http://ansambl.szm.sk/TEXTY/MP3/index.htm.

[2] *Audio coding format* [online]. [cit. 2021-12-4]. Dostupné z: https://en.wikipedia.org/wiki/Audio_coding_format.

[3] *Biofyzika sluchu* [online]. [cit. 2021-11-20]. Dostupné z: https://www.wikiskripta.eu/w/Biofyzika_sluchu.

[4] *Fourierova transformácia* [online]. [cit. 2021-3-21]. Dostupné z: https://sk.wikipedia.org/wiki/Fourierova_transform%C3%A1cia.

[5] *MP3* [online]. [cit. 2021-12-4]. Dostupné z: https://en.wikipedia.org/wiki/MP3.

[6] *Sampling (signal processing)* [online]. [cit. 2021-12-4]. Dostupné z: https://en.wikipedia.org/wiki/Sampling_(signal_processing).

[7] *Steely Dan* [online]. [cit. 2021-12-4]. Dostupné z: https://en.wikipedia.org/wiki/Steely_Dan.

[8] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Dostupné z: https://www.tensorflow.org/.

[9] AcousticBrainz. *MSD Rosetta API Query Dataset* [online]. 2016 [cit. 2022-1-6]. Dostupné z: https://ftp.acousticbrainz.org/pub/acousticbrainz/acousticbrainz-labs/download/msdrosetta/.

[10] Bertin Mahieux, T., Ellis, D. P., Whitman, B. a Lamere, P. The Million Song Dataset. In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*. 2011.

[11] Defferrard, M., Benzi, K., Vandergheynst, P. a Bresson, X. FMA: A Dataset for Music Analysis. In: *18th International Society for Music Information Retrieval Conference (ISMIR)*. 2017. Dostupné z: https://arxiv.org/abs/1612.01840.

[12] Deshpande, A. *A Beginner's Guide To Understanding Convolutional Neural Networks*. 2019 [cit. 2022-2-23]. Dostupné z: https://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/.

[13] Durčák, I. P. *Neuronové sítě a princip jejich fungování*. 2017 [cit. 2022-2-17]. Dostupné z: https://www.napocitaci.cz/33/neuronove-site-a-princip-jejich-fungovani-uniqueidgOkE4NvrWuNY54vrLeM670eFNQh552VdDDulZX7UDBY/.

[14] Education, I. C. *Machine Learning* [online]. 2020 [cit. 2022-2-16]. Dostupné z: https://www.ibm.com/cz-en/cloud/learn/machine-learning.

[15] Flanagan, D. *JavaScript: The Definitive Guide*. 6. vyd. O'Reilly Media, Inc., 2011. ISBN 978-0-596-80552-4.

[16] Gurucharan, M. *Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network*. December 2020 [cit. 2022-3-13]. Dostupné z: https://www.upgrad.com/blog/basic-cnn-architecture/.

[17] Hosken, D. *An Introduction to Music Technology*. 1. vyd. Routledge, 2011. ISBN 0-203-84951-5.

[18] Ioffe, S. a Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Marec 2015. Dostupné z: https://arxiv.org/abs/1502.03167.

[19] Kozlová, D. *Detekce anatomických struktur v CT datech s využitím konvolučních neuronových sítí*. Brno, CZ, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních techologií. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=171750.

[20] Krajíček, J. *Tvorba spektrogramů a jejich zpětná syntéza*. Praha, CZ, 2010. Bakalářská práce. Univerzita Karlova v Praze, Matematicko-fyzikální fakulta. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=171750.

[21] Lachmish, M. *MusicGenreClassification* [online]. 2016 [cit. 2022-1-6]. Dostupné z: https://github.com/mlachmish/MusicGenreClassification.

[22] Li, X., Chen, S., Hu, X. a Yang, J. Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift. Január 2018. Dostupné z: https://arxiv.org/abs/1801.05134.

[23] McFee, Brian, Raffel, C., Liang, D., Ellis, D. P. et al. librosa: Audio and music signal analysis in python. In: *14th python in science conference*. 2015. Dostupné z: http://conference.scipy.org/proceedings/scipy2015/pdfs/brian_mcfee.pdf.

[24] Mishkin, D., Sergievskiy, N. a Matas, J. Systematic evaluation of convolution neural network advances on the Imagenet. *Computer Vision and Image Understanding*. 2017. DOI: https://doi.org/10.1016/j.cviu.2017.05.007. ISSN 1077-3142. Dostupné z: http://www.sciencedirect.com/science/article/pii/S1077314217300814.

[25] Moujahid, A. *A Practical Introduction to Deep Learning with Caffe and Python*. 2016 [cit. 2022-2-16]. Dostupné z: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/.

[26] Muráň, I. J. *Úvod do konvolučných neurónových sietí*. 2019 [cit. 2022-2-23]. Dostupné z: https://umelainteligencia.sk/uvod-do-konvolucnych-neuronovych-sieti/.

[27] Olteanu, A. *GTZAN* [online]. 2019 [cit. 2021-12-8]. Dostupné z: https://www.tensorflow.org/datasets/catalog/gtzan.

[28] Olteanu, A. *GTZAN Kaggle* [online]. 2019 [cit. 2021-12-8]. Dostupné z: https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification.

[29] Oujezdský, A. *Zpracování zvuku na počítači*. 1. vyd. Ostravská univerzita v Ostravě, 2014 [cit. 2021-11-18]. Dostupné z: https://docplayer.cz/18192418-Zpracovani-zvuku-na-pocitaci.html.

[30] Schreiber, H. The Million Song Dataset. In: *Improving Genre Annotations for the Million Song Dataset Retrieval (ISMIR 2015)*.

[31] Shang, W., Sohn, K., Almeida, D. a Lee, H. Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units. Marec 2016.

[32] Singh, V. *Flask vs Django in 2022: Which Framework to Choose?* Január 2022 [cit. 2022-4-3]. Dostupné z: https://hackr.io/blog/flask-vs-django.

[33] Tzanetakis, G., Essl, G. a Cook, P. *Automatic Musical Genre Classification Of Audio Signals*. The International Society for Music Information Retrieval, 2001. Dostupné z: http://ismir2001.ismir.net/pdf/tzanetakis.pdf.

[34] Woodford, C. *Microphones* [online]. 2021 [cit. 2021-11-20]. Dostupné z: https://www.explainthatstuff.com/microphones.html.

[35] Yegulalp, S. *What is TensorFlow? The machine learning library explained*. Jún 2019 [cit. 2022-4-3]. Dostupné z: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html.

[36] Zbořil, F. V. *Základy umělé inteligence*. Fakulta informačních technologií VUT v Brně, 2022 [cit. 2022-2-16]. Dostupné z: https://www.fit.vutbr.cz/study/courses/IZU/private/2022-opora-IZU.pdf.