Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Automatizovaná transkripce hudebních nahrávek do notového zápisu

Bakalářská práce

2024 Tomáš Netolický

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš Netolický**
Osobní číslo:	**I22021**
Studijní program:	**B0714A150008 Automatizace**
Téma práce:	**Automatizovaná transkripce hudebních nahrávek do notového zápisu**
Zadávající katedra:	**Katedra automatizace a matematiky**

Zásady pro vypracování

Cílem této bakalářské práce je navrhnout a implementovat systém pro automatickou transkripci hudebních nahrávek do notového zápisu. Systém by měl zpracovat audio soubory vhodnými běžnými metodami v kombinaci s metodami strojového učení pro rozpoznání jednotlivých hudebních nástrojů a převést hudební obsah do notového zápisu. V teoretické části student provede rešerši v oblasti digitálního zpracování hudebního signálu a transkripce zvuku do notového zápisu. Důraz bude kladen na metody analýzy zvukového spektra a principy strojového učení, které umožňují klasifikaci hudebních nástrojů na základě zvukového vzorku.
V praktické části student navrhne a implementuje program v Pythonu, který bude schopen načítat audio soubory a pomocí použitých metody generovat odpovídající notové zápisy. Výstup bude možné uložit ve standardním formátu. Kromě samotného programu bude proveden a popsán návrh metody strojového učení pro řešení problému transkripce samotné, a to včetně popisu tvorby datových sad, trénování, validace a testování výsledných modelů. V závěru práce student zhodnotí funkčnost a přesnost transkripce, analyzuje úspěšnost rozpoznávání hudebních nástrojů a diskutuje o možných vylepšeních v rámci rozpoznávání a přesnosti zápisu.

Rozsah pracovní zprávy: **cca 40 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Signal Processing and Machine Learning Theory, 2024. Online. Elsevier. ISBN 9780323917728. Available at: https://doi.org/10.1016/C2021-0-01229-3.
LERCH, Alexander, [2012]. An introduction to audio content analysis: applications in signal processing and music informatics. Piscataway, NJ: IEEE Press. ISBN 978-1-118-26682-3.
CORD, Matthieu a CUNNINGHAM, Pádraig, 2008. Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval. Berlin: Springer-Verlag Berlin and Heidelberg GmbH & Co. KG. ISBN 354075170X.

Vedoucí bakalářské práce: **Ing. Dominik Štursa, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2024**
Termín odevzdání bakalářské práce: **16. května 2025**

L.S.

prof. Ing. Petr Doležel, Ph.D. v.r.
děkan

Ing. Libor Kupka, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 24. ledna 2025

Prohlašuji:

Práci s názvem Automatizovaná transkripce hudebních nahrávek do notového zápisu jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 9. května 2025

Tomáš Netolický v. r.

PODĚKOVÁNÍ

Na tomto místě bych rád poděkoval vedoucímu bakalářské práce Ing. Dominiku Štursovi, Ph.D. za vstřícnost, konstruktivní zpětnou vazbu a odborný pohled, který mi pomohl lépe uchopit téma a dostat práci do finální podoby. Dále děkuji své rodině, přátelům a přítelkyni za jejich podporu, porozumění a motivaci v průběhu celého studia. Zvláštní poděkování patří také mému učiteli hry na trumpetu Bc. Jaroslavu Vašíčkovi, který mě dlouhodobě inspiroval a ovlivnil můj vztah k hudbě.

ANOTACE

Tato bakalářská práce se zabývá automatizovanou transkripcí hudebních nahrávek do notového zápisu s využitím metod strojového učení. V teoretické části jsou popsány základy hudební teorie, principy frekvenční analýzy, reprezentace zvukového signálu a metody hlubokého učení využívané pro extrakci hudebních prvků. Praktická část se zaměřuje na návrh a implementaci neuronové sítě, která na základě konstantní-Q transformace (CQT) a krátkodobé Fourierovy transformace (STFT) identifikuje výšku a délku tónů. Výsledky jsou zpracovány do formátu MIDI pro další využití v hudební analýze a produkci.

KLÍČOVÁ SLOVA

frekvenční analýza, neuronové sítě, transkripce hudby, binární klasifikace

TITLE

Automated transcription of musical recordings into sheet music

ANNOTATION

This bachelor's thesis focuses on the automated transcription of musical recordings into sheet music using machine learning methods. The theoretical part describes the principles of frequency analysis, sound signal representation, and deep learning techniques used for extracting musical elements. The practical part is dedicated to the design and implementation of a neural network that identifies pitch and note duration based on the Constant-Q Transform (CQT) and Short-Time Fourier Transform (STFT). The results are processed into the MIDI format for further use in music analysis and production.

KEYWORDS

frequency analysis, neural networks, music transcription, binary classification

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM UKÁZEK KÓDU

SEZNAM ZKRATEK A ZNAČEK

SR	Vzorkovací frekvence, Sample rate
FT	Fourierova transformace, Fourier transform
DFT	Diskrétní Fourierova transformace, Discrete Fourier transform
FFT	Rychlá Fourierova transformace, Fast Fourier transform
STFT	Krátkodobá Fourierova transformace, Short-Time Fourier transform
CQT	Konstantní-Q transformace, Constant-Q transform
HCQT	Harmonická konstantní-Q transformace, Harmonic constant-Q transform
NN	Neuronová síť, Neural network
DL	Hluboké učení, Deep learning
MLP	Vícevrstvý perceptron, Multilayer perceptron
CNN	Konvoluční neuronová síť, Convolutional neural network
RNN	Rekurentní neuronová síť, Recurrent neural network

ÚVOD

Tato bakalářská práce se zabývá návrhem a realizací automatizovaného systému pro transkripci hudebních nahrávek do notového zápisu pomocí metod strojového učení. Výsledkem by měl být model schopný identifikovat výšku a délku jednotlivých tónů ve skladbě a převádět je do formátu MIDI, který lze dále využít pro hudební analýzu a produkci. Model bude pracovat s frekvenční analýzou zvukového signálu a využívat konstantní-Q transformaci (CQT), její o oktávu a duodecimu rozšířenou obdobu harmonickou konstantní-Q transformaci (HCQT) a krátkodobou Fourierovu transformaci (STFT) jako vstupní reprezentaci dat. Výstup reprezentuje binární matice hodnot.

Teoretická část práce se zaměří na popis metod frekvenční analýzy, které se používají při zpracování hudebního signálu. Dále budou představeny principy strojového učení a hlubokých neuronových sítí, které umožňují efektivní rozpoznávání tónů v hudebních nahrávkách. Bude zmíněna i hudební teorie pro lepší pochopení dané problematiky.

Praktická část práce bude věnována návrhu a implementaci modelu pro automatickou transkripci. Bude popsán použitý dataset, proces přípravy vstupních a výstupních dat, volba architektury neuronové sítě a optimalizace parametrů pro dosažení co nejpřesnějších výsledků. Dále bude popsána konverze výstupu modelu do formátu MIDI a experimentální vyhodnocení přesnosti transkripce pomocí volně přístupného programu pro vytváření a editaci hudebních zápisů MuseScore Studio 4.

1 TEORETICKÁ ČÁST

1.1 Základní hudební pojmy a jejich význam pro digitální zpracování

1.1.1 Základy hudební teorie

Tón je základním stavebním kamenem každé hudební skladby. Jeho vlastnosti, jako jsou výška, délka, intenzita a barva, určují charakter hudebního projevu. Kombinací tónů vznikají melodie, harmonie i rytmické struktury, které tvoří hudební celek.

Základní fyzikální vlastností tónu je jeho výška neboli frekvence (vyjádřená v hertzech, Hz). Čím vyšší je frekvence, tím vyšší je vnímaný tón. V hudební teorii se tóny řadí do oktáv, které se skládají vždy z 12 tónů s pevně danými frekvencemi. Těmito tóny jsou: c, c# (cis), d, d# (dis), e, f, f# (fis), g, g# (gis), a, a# (ais) a h. Každý zvýšený tón v tomto výčtu má také svůj enharmonický ekvivalent – například f# (fis) odpovídá frekvencí tónu g♭ (ges). Tyto noty zní totožně, ale jejich použití se liší v závislosti na hudebním kontextu.

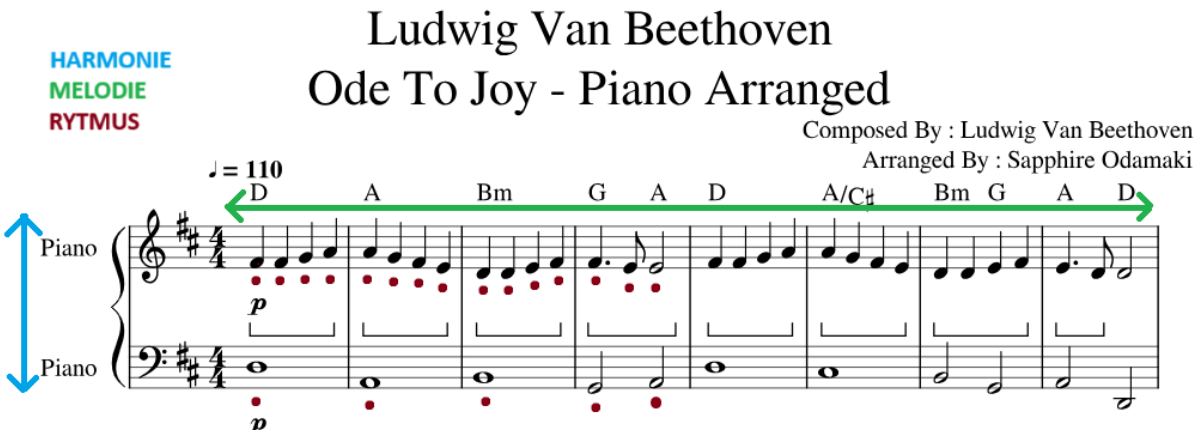
I když se v temperovaném ladění pracuje s dvanácti základními tóny (c, c♯/d♭, d, d♯/e♭ atd.), hudební teorie umožňuje i existenci dalších tónů, jako jsou například c𝄪 (cisis) nebo g𝄫 (geses). Tyto tóny se vyskytují zejména v některých hudebních kontextech, například v mikrotonální hudbě nebo při teoretické analýze enharmonických vztahů. Detailnější rozbor těchto jevů však přesahuje rámec této práce.

Melodie je tvořena posloupností tónů uspořádaných v čase (v tempu). Může být jednoduchá, skládající se z několika tónů, nebo složitá s rozsáhlými variacemi. Každý jednotlivý hlas skladby (nástroj) má svou vlastní melodii. Rozeznání melodie je klíčové pro automatizovanou transkripci hudby.

Harmonie vzniká současným zněním několika tónů najednou. Z teoretického hlediska je harmonie založena na vztazích mezi tóny a jejich souzvucích. Automatizovaná transkripce musí být schopna rozlišit jednotlivé souzvuky a odhalit, z jakých tónů se akordy skládají.

Rytmus je organizace zvuků v čase. Tvoří základní strukturu hudby. Rytmus určuje, kdy jednotlivé tóny začnou a jak dlouho budou znít. Rytmus je vytvářen pomocí různých tónových délek a pomlk (nota/pomlka – celá, půlová, čtvrťová, osminová, …).

Na následujícím obrázku je na skladbě Óda na radost od Ludwiga Van Beethovena, aranžér Sapphire Odamaki, naznačeno, kde se projevuje harmonie, melodie a rytmus skladby.

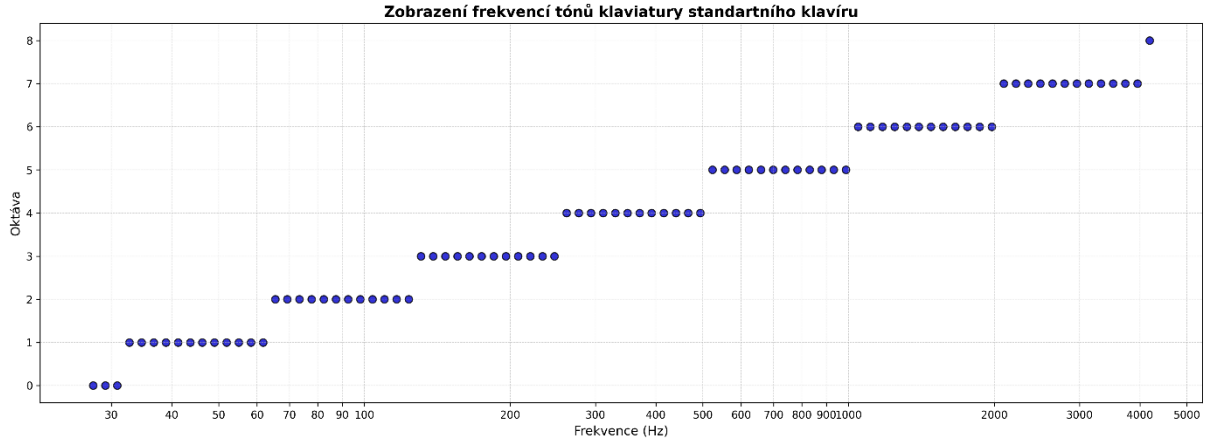


Obrázek 1 - ukázka harmonie, melodie a rytmu na skladbě Óda na radost, autor skladby: Ludvig Van Beethoven, aranžmá: Sapphire Odamaki, upravil: autor (Sapphire Odamaki, 2025)

Podle (Hons, 2019) má harmonie velice úzký vztah k melodii. I neakordické tóny (tóny, které nejsou součástí základního akordu) ovlivňují vztah mezi melodií a harmonií. Rytmus je považován za klíčový prvek ve formování hudební struktury společně s melodií a harmonií a hraje velice důležitou roli při utváření hudební formy.

1.1.2 Frekvenční rozložení tónů

Jak již bylo zmíněno, jednou ze základních vlastností tónu je jeho výška, tedy frekvence. Tóny se na frekvenční ose nevyskytují rovnoměrně, ale v logaritmickém měřítku. Pro lidské ucho však působí jako rovnoměrně rozprostřené – každý půltón vnímáme jako stejně velký krok, a to díky přibližně logaritmickému charakteru lidského sluchu. Z tohoto důvodu si můžeme představit tónovou řadu jako ekvidistantně rozdělenou, jak je znázorněno na obrázku 2.



Obrázek 2 - graf rozložení tónů standartní klaviatury v logaritmickém frekvenčním měřítku (autor, 2025)

Frekvence jednotlivých tónů určuje nejenom výšku, ale i jejich barvu. Lidskému uchu mohou tóny nízkých frekvencí připadat hutné, pomalé, zatímco tóny vyšších frekvencí mohou člověku připadat rychlejší a ostřejší.

V následující tabulce je přehled frekvencí některých tónů. Modrošedé políčko označuje tón a^1 o frekvenci 440 Hz, který se používá jako referenční tón pro ladění (nastavení zvuku nástroje tak, aby výška jednotlivých tónů odpovídala ostatním nástrojům) nejen symfonických orchestrů.

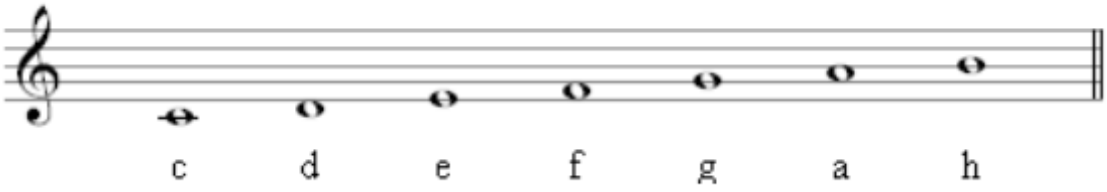
Tabulka 1 - ukázka frekvencí not v prvních 9 oktávách (autor, 2025)

TÓNY	OKTÁVA								
	0	1	2	3	4	5	6	7	8
c	16,35	32,70	65,41	130,81	261,63	523,25	1 046,50	2 093,00	4 186,01
cis	17,32	34,65	69,30	138,59	277,18	554,37	1 108,73	2 217,46	4 434,92
d	18,35	36,71	73,42	146,83	293,66	587,33	1 174,66	2 349,32	4 698,64
dis	19,45	38,89	77,78	155,56	311,13	622,25	1 244,51	2 489,02	4 978,03
e	20,60	41,20	82,41	164,81	329,63	659,26	1 318,51	2 637,02	5 274,04
f	21,83	43,65	87,31	174,61	349,23	698,46	1 396,91	2 793,83	5 587,65
fis	23,12	46,25	92,50	185,00	369,99	739,99	1 479,98	2 959,96	5 919,91
g	24,50	49,00	98,00	196,00	392,00	783,99	1 567,98	3 135,96	6 271,93
gis	25,96	51,91	103,83	207,65	415,30	830,61	1 661,22	3 322,44	6 644,88
a	27,50	55,00	110,00	220,00	440,00	880,00	1 760,00	3 520,00	7 040,00
ais	29,14	58,27	116,54	233,08	466,16	932,33	1 864,66	3 729,31	7 458,62
h	30,87	61,74	123,47	246,94	493,88	987,77	1 975,53	3 951,07	7 902,13

V moderní hudbě už se od referenčního ladění na frekvenci 440 Hz upouští a přechází se na 442–444 Hz. Někteří dirigenti a skladatelé experimentují s laděním na 432 Hz, což je podle některých „přirozené ladění“.

1.1.3 Notové zápisy a hudební klíče

V předchozích kapitolách bylo zmíněno, že hudební systém je založen na opakujícím se sledu sedmi základních (dvanácti celkem) tónů, známých také jako hudební abeceda. Tyto tóny tvoří základní stavební prvek veškeré západní hudby. Jakmile dosáhneme konce této řady, názvy tónů se opakují ve vyšších či nižších polohách. Aby bylo možné mezi stejně pojmenovanými tóny rozlišovat jejich výšku, označují se navíc podle oktávy, ve které se nacházejí. Oktáva představuje vzdálenost mezi dvěma tóny stejného jména – například mezi c^1 a c^2.

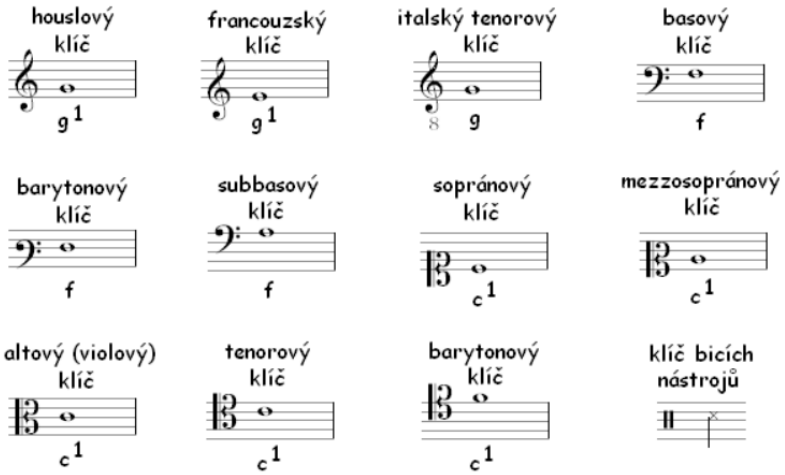


Obrázek 3 - základní tónová řada zapsána v notové osnově (Profous, 2011)

Na začátku notové osnovy na obrázku 3 je tzv. hudební klíč, v tomto případě houslový klíč. Hudební klíč je značka, která určuje, kde se v notové osnově nachází konkrétní tón – například g^1, f^1 nebo c^1, podle typu použitého klíče. Od této výchozí pozice se pak odvíjí umístění všech ostatních tónů. Teoreticky lze vytvořit až 45 různých klíčů – kombinací tří základních typů klíčů, pěti linek osnovy, na které je možné klíč umístit, a tří variant transpozice (běžná, o oktávu výš nebo o oktávu níž, přičemž transpozice se označuje číslicí 8 nad nebo pod klíčem).

Existují tři hlavní typy klíčů, vyobrazené na obrázku 4, které se v současnosti používají:

- G-klíč (nejčastěji houslový klíč) označuje, kde se nachází tón g^1. Tento klíč se zapisuje tak, že jeho spirála začíná na lince, kde je tón umístěn. Nejznámější variantou je houslový klíč, který umísťuje tón g^1 na druhou linku (linky se počítají odspodu). Houslový klíč se používá pro nástroje s vyšším laděním, jako je flétna, trumpeta nebo housle.
- F-klíč vyznačuje umístění tónu f (malé f), který se nachází mezi dvěma tečkami tvořícími klíč. Nejznámější variantou je basový klíč, ve kterém je tento tón umístěn na čtvrté lince. Basový klíč se používá pro nástroje s hlubším laděním, jako je kontrabas.
- C-klíč (nejčastěji altový/violový klíč) označuje pozici tónu c^1. Ten se nachází na lince, kterou v klíči označuje středový „zobáček“. Tyto klíče se historicky používaly především pro vokální hlasy. Dnes se s nimi setkáme hlavně u violy, případně v některých vysokých polohách violoncella a pozounu.



Obrázek 4 - tabulka hudebních klíčů vč. houslového, basového a altového (violového) klíče (Profous, 2011)

Podle (Profous, 2011).

1.1.4 Vznik tónu a tónové rozsahy hudebních nástrojů

Hudební nástroje můžeme mimo jiné dělit i podle toho, jak se na něj hraje neboli jak se na tyto nástroje vytváří tóny. Podle této klasifikace (Profous, 2011) se hudební nástroje dělí na následující skupiny.

Nástroje strunné:

- smyčcové (housle, violoncello, kontrabas)
- drnkací (kytara, banjo, harfa, loutna)
- úderné (klavír, pianino, cembalo)

Nástroje dechové:

- dřevěné (zobcová a příčná flétna, klarinet, saxofon, hoboj)
- žesťové (trumpeta, trombon, lesní roh, tuba, eufonium)
- vícehlasé (varhany, akordeon, dudy, foukací harmonika)

Nástroje bicí:

- samozvučné (xylofon, marimba, zvonkohra)
- blanozvučné (tympány, bonga, conga, velký a malý buben)

Již podle této klasifikace lze odhadnout, jak které nástroje vytváří tón. Nyní bude uvedeno několik nejznámějších a nejpoužívanějších nástrojů a u každého z nich také způsob, jakým vytváří tón. Budou také zmíněny přibližné rozsahy jednotlivých nástrojů, aby bylo čtenáři přiblíženo, co lze na jaký nástroj zahrát.

Strunné smyčcové nástroje (housle, violoncello, kontrabas, …)

Hráč na housle, potažmo další strunné smyčcové nástroje, vytváří tón taháním smyčce vyrobeného z koňských žíní po strunách. Tento pohyb, ať už je plynulý, dlouhý anebo krátký či skákavý, rozkmitá jednotlivé struny a tím vytvoří žádaný tón.

Chromatický rozsah houslí na obrázku 5 je přibližně 4 oktávy od malého g do g^4, přičemž jednotlivé struny houslí jsou laděny kvintově. Jedná se o tóny g (196 Hz), d^1 (293,7 Hz), a^1 (440 Hz) a e^2 (659,3 Hz). Housle se notují v houslovém (G) klíči.

Obrázek 5 - housle a smyčec (Profous, 2011)

Strunné úderné nástroje (klavír, piano, pianino, cembalo)

Úderné strunné nástroje jako je klavír nebo piano vytváří tón úderem kladívka do struny. Každá klávesa klaviatury nástroje má své kladívko a svou strunu, pomocí které vytváří tón. Délku a hlasitost hraného tónu je možné ovlivnit až pomocí 3 pedálů (počet pedálů závisí na nástroji).

Chromatický rozsah klavíru na obrázku 6 je téměř 8 oktáv, od A_2 do c^5, tedy 88 tónů/kláves. Klaviatura klavíru se dělí na bílé a černé klávesy, kde bílé klávesy jsou základní tóny stupnice C dur (c, d, e, f, …, h, c) a černé klávesy představují zvýšené/snížené tóny (cis/des, dis/es, …).

Rozsahy dalších nástrojů z této kategorie jsou: piano a pianino – A_2 až c^5, cembalo – F_1 až f^3 nebo G_1 až d^3.

Obrázek 6 - klavír (Profous, 2011)

Dechové dřevěné nástroje (flétna, zobcová flétna, příčná flétna)

Na zobcovou flétnu vzniká tón pomocí dopravení vzduchu do tzv. zobce. Pomocí otvorů na vrchní straně může hráč měnit výšku tónu. U příčné flétny tón vzniká dopravením vzduchu do nástroje přes hranu kruhového nebo čtyřhranného otvoru. Výška tónu je ovládána pomocí klapek, které určí výšku vzduchového sloupce. (Tvrz, 2021)

Rozsahově se flétny liší podle typu flétny, zde jsou přibližné rozsahy těch nejpoužívanějších: zobcová sopránová (nejčastěji používaná ve výuce v základních uměleckých školách, obrázek 7 vlevo) – c^1 až d^3, zobcová altová – f až g^2, koncertní příčná flétna (obrázek 7 vpravo) – c^1 až d^4.

Obrázek 7 - zobcová (vlevo) a příčná (vpravo) flétna (Tvrz, 2021)

Dechové plátkové nástroje (klarinet, saxofon)

Klarinet a saxofon patří mezi dřevěné plátkové dechové nástroje, na které hráč vytváří tón pomocí rozechvění třtinového plátku umístěného v hubičce nástroje. Výšku tónu potom hráč upravuje pomocí otvorů a klapek po celé délce nástroje. Rozdíl mezi těmito nástroji je ve tvaru, a hlavně v materiálu, ze kterého je vyrobený. I když saxofon patří mezi dřevěné nástroje, je zásadně vyráběn z kovu, přesněji z mosazi.

Běžný rozsah klarinetu je od malého d do c^3. Samozřejmě profesionální hráči mají většinou mnohem větší rozsahy, což platí u všech hudebních nástrojů.

U saxofonů závisí rozsah na typu nástroje. Mezi nejčastěji používané saxofony patří: sopránový, altový, tenorový a barytonový. Dále se lze setkat i se saxofony sopraninovými nebo kontrabasovými. Pro nejpoužívanější nástroje se dají rozsahy vyjádřit následovně: altový saxofon – malé des až as^2, tenorový saxofon – velké As až des^2, barytonový saxofon na obrázku 8 – velké Des až ges^1. (Tvrz, 2021)

Obrázek 8 - baryton saxofon (Paul Effman Music, 2025)

Žesťové nátrubkové nástroje (trumpeta, trombon)

Tento druh dechových nástrojů je vyráběn ze slitiny mědi a zinku, která se nazývá mosaz. Vzduch se do těchto nástrojů dopravuje pomocí nátrubku, který má hráč při hraní přitisknutý na rtech. Hráč do nátrubku nejen že vydechuje vzduch, ale musí i vytvořit speciální typ zvuku, tzv. „bzučení“. Velikosti nátrubků se odvíjejí od velikosti nástroje.

Trumpeta patří k nejznámějším žesťovým nástrojům. Výška tónu se mimo nátisku, způsobu přiložení a napnutí rtů na nátrubku, koriguje pomocí tří pístů, které lze vidět na obrázku 9. Trumpeta má 2^3 různých hmatů, pomocí kterých se koriguje vzduchový sloupec. Rozsah běžného hráče na trumpetu je od malého g do g^2, přičemž pokročilí až profesionální hráči mohou hrát g^3 a výš.

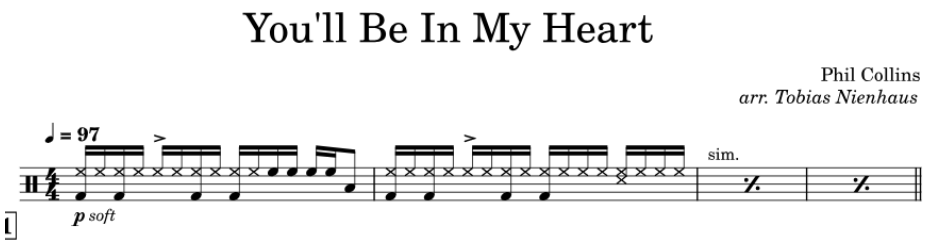
Pozouny se dělí na snižcové a ventilové. Ventilové pozouny fungují na podobném principu jako trumpeta, u snižcového pozounu se tón určuje polohou snižce. Rozsah nástroje se pohybuje od kontra B_1 do d^2.

Obrázek 9 - trumpeta (Paul Effman Music, 2025)

Bicí nástroje (bicí souprava, tympány)

Na rozdíl od zmíněných nástrojů, bicí nástroje nemají konkrétní tónovou výšku. Notový zápis u bicích nástrojů neslouží k určení výšky tónu, ale k rozlišení konkrétního bubnu nebo činelu, na který má být v okamžiku zahráno (názorná ukázka na obrázku 11) a v případě bicí soupravy na jaký buben (viz. obrázek 10). Z předchozího textu je zřejmé, že u bicích nástrojů nelze určit rozsah.

Obrázek 10 - bicí souprava (Profous, 2011)



Obrázek 11 - ukázka notace pro bicí soupravu (Tobias.nienhaus, 2025)

1.1.5 Význam MIDI v moderní hudbě

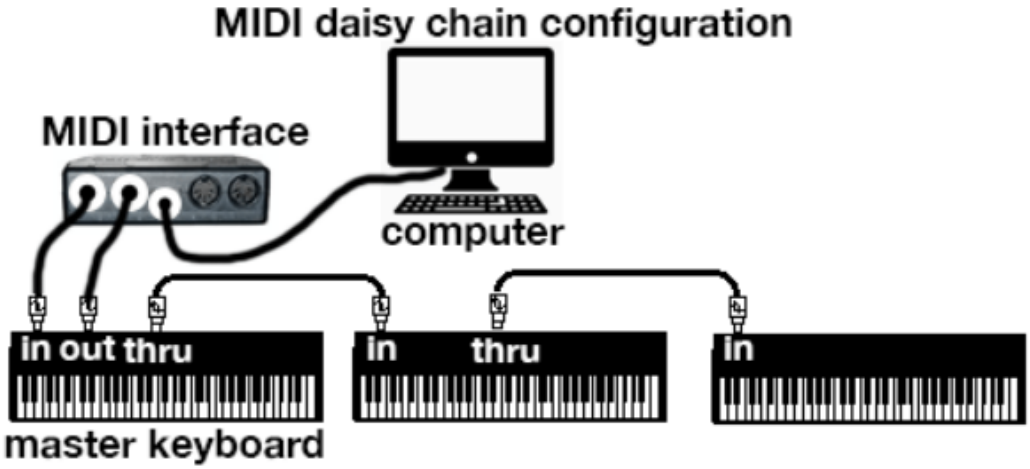
Zkratka MIDI znamená Musical Instrument Digital Interface, tedy digitální rozhraní pro hudební nástroje. Jedná se o komunikační protokol, který umožňuje propojení různých hudebních zařízení a jejich vzájemnou výměnu dat. Tento standard se rozšířil zejména v oblasti digitální hudby a slouží jako prostředník mezi nástroji, sekvencery a počítači.

Ačkoli MIDI přenáší technicky specifické informace ve formě čísel a kódů, pro běžnou práci s hudebním softwarem není nutné znát jejich přesnou strukturu. Důležité je pochopit základní princip – že MIDI neslouží k přenosu zvuku samotného, ale informací o tom, co a jak má být přehráno.

Z pohledu techniky funguje MIDI jako sériová forma přenosu dat, kde se přenášejí pouze změny stavů určitých parametrů. Když například hudebník stiskne klávesu, vyšle zařízení zprávu označenou jako Key On. Během držení klávesy se žádné další informace neposílají,

dokud nedojde k jejímu uvolnění – v tu chvíli se vyšle zpráva Key Off. Hudební zařízení si mezitím pamatují poslední přijatý stav, dokud neobdrží novou instrukci.

Aby bylo možné vést více nezávislých komunikací současně přes jeden MIDI kabel, je protokol rozdělen do 16 samostatných kanálů. Každá zpráva začíná informací o tom, kterého kanálu se týká, a teprve poté následují konkrétní datové hodnoty. Přes jeden MIDI řetězec tedy mohou procházet všechny zprávy najednou, jak je vidět na zapojení na obrázku 12 mezi elektrickými klávesami, ale každý nástroj reaguje pouze na ty, které jsou určeny jemu – tedy na zprávy v kanálu, na který je nastaven. (Jež, 2017)



Obrázek 12 - možné zapojení počítače, MIDI rozhraní a el. kláves v tzv. daisy chain configuration (Hass, 2025)

V rámci této práce hraje MIDI hlavní roli v praktické části díky své schopnosti uchovávat hudební data ve formátu .mid. Tyto soubory obsahují sekvence MIDI zpráv, které popisují parametry jednotlivých tónů, jako je výška (pitch), intenzita/dynamika (velocity), čas zapnutí a vypnutí noty a také informace o kanálu a o programu. Program má význam celých čísel, přičemž každé číslo odpovídá nástroji, který má danou notu zahrát (např. 0 – klavír, 74 – flétna).

Pro účely této práce je důležité zmínit to, že při práci s .mid soubory pracuje uživatel s hudbou tzv. časovým přístupem. Tento přístup umožňuje uživateli zaznamenat noty pomocí času začátku a konce a výšky. Například notu a^1 můžeme zaznamenat: begin = 1,256 s, end = 1,587, pitch = 69 (podle standartního MIDI číslování). Oproti tomu pomocí klasického hudebního přístupu je nutné stejnou notu zaznamenat pomocí následujících hodnot: pitch = a_1, délka = osmina, takt = 7.

1.2 Digitální reprezentace zvuku

1.2.1 Zvuk jako analogový signál

Vzduch, který dýcháme, je pružné médium – dokáže se přizpůsobit změnám a poté se vrátit do původního rovnovážného stavu. Díky této vlastnosti vzniká zvuk: molekuly vzduchu jsou narušeny a uvedeny do pohybu. K tomu dochází tehdy, když začne kmitat nějaký objekt označovaný jako zdroj zvuku. Molekuly vzduchu kolem tohoto zdroje narážejí do dalších molekul, čímž vznikají oblasti s vyšším a nižším tlakem, které se šíří od zdroje dál ve formě akustických vln.

Naše uši tyto změny tlaku vnímají a přenášejí je do mozku, kde jsou interpretovány jako zvuk. Přenos zvuku tedy zahrnuje tři hlavní části:

- zdroj zvuku, který kmitá – například struna houslí,
- prostředí, kterým se tyto vibrace šíří – nejčastěji vzduch, ale může to být i voda nebo jiná látka,
- příjemce zvuku, který zachytí změny tlaku ve vzduchu (nebo jiném médiu) a umožní jejich zpracování.

V tradiční hudbě hudebník přímo manipuluje s nástrojem, aby vytvořil požadovaný zvuk. U různých nástrojů to probíhá různě – pianista stiskne klávesy, houslista táhne smyčcem po struně, perkusionista udeří do bubnu. Ve všech případech hudebník přímo ovlivňuje zdroj zvuku, tedy něco, co mechanicky vibruje.

V elektronické hudbě je ale konečným zdrojem zvuku reproduktor, který vibruje a vytváří akustickou energii. Hudebníci ho však nijak fyzicky neovlivňují – neudeří ho ani po něm netáhnou smyčcem. Místo toho pracují s reprezentací zvuku, tedy s jeho záznamem nebo zpracováním v jiné formě. Právě tato práce s reprezentacemi je podstatou elektronické hudby.

Zvuk může být zachycen a převeden na elektrickou podobu – například mikrofonem, který akustické vlnění (tedy změny tlaku ve vzduchu) převede na kolísající elektrické napětí. Výsledkem je tzv. analogový signál, který odpovídá původnímu zvuku. Tento analogový signál lze dále zesílit a přivést zpět do reproduktorů, které ho znovu promění na akustický zvuk. Nebo může být uložen na magnetický pásek jako analogová nahrávka. Přehráním pásku se proces obrátí – uložená data se změní zpět na elektrický analogový signál. (Stolet, 2009)

1.2.2 Digitalizace a rekonverze zvuku

Analogový signál lze převést do digitální podoby pomocí zařízení zvaného analogově-digitální převodník (ADC – Analog-to-Digital Converter). Tento převodník převede plynulé změny napětí, které odpovídají zvukovým vlnám, na posloupnost čísel – tedy na digitální reprezentaci zvuku. Jakmile je zvuk reprezentován digitálně, může být dále upravován, zpracován, nebo uložen ve formě digitální nahrávky.

Naopak pro návrat do analogové podoby se používá digitálně-analogový převodník (DAC – Digital-to-Analog Converter). DAC převádí čísla zpět na napěťové signály, které lze následně zesílit a přehrát pomocí reproduktorů jako akustický signál.

Zvuk tedy může být reprezentován dvěma způsoby – analogově, což znamená plynulé kolísání napětí, nebo digitálně, kde jsou amplitudy zaznamenávány jako jednotlivé vzorky v čase. Proces převedení analogového signálu do digitální podoby se nazývá digitální vzorkování.

Během vzorkování jsou v pravidelných časových intervalech měřeny okamžité hodnoty amplitudy signálu. Každé takové měření se nazývá vzorek (sample). Tyto vzorky tvoří digitální podobu signálu, která sice není přesnou kopií originálu, ale při dostatečně hustém vzorkování ho dokáže velmi dobře přiblížit.

Dva klíčové pojmy, které při digitalizaci hrají roli, jsou:

- Vzorkovací frekvence (sampling rate, SR) – udává, kolik měření se provede za jednu sekundu. Běžné hodnoty jsou 44 100 Hz nebo 48 000 Hz. Podle Nyquistova kritéria musí být vzorkovací frekvence alespoň dvojnásobkem nejvyšší frekvence, kterou chceme zaznamenat, jinak může dojít k tzv. aliasingu, tedy k deformaci zvuku způsobené falešnými frekvencemi.
- Rozlišení (bitová hloubka) – určuje, do kolika různých úrovní může být každé měření amplitudy zařazeno. Například 8bitový systém umožňuje 256 různých hodnot, 16bitový pak 65 536 různých hodnot, vyšší rozlišení znamená menší zaokrouhlování, a tedy přesnější digitální reprezentaci původního zvuku.

Digitalizace a rekonverze zvuku spolu také přinášejí některá úskalí, kterým je v digitální hudbě nutné čelit a aktivně řešit či potlačovat. Mezi tyto problémy patří již dříve zmíněný aliasing, kvantizační šum či jiná dynamická a rozsahová zkreslení.

Kvantizační šum

Při převodu analogového signálu na digitální dochází ke kvantizaci – tedy převodu spojitých hodnot napětí na diskrétní digitální úrovně. Tento proces zahrnuje zaokrouhlování skutečných hodnot na nejbližší možnou digitální hodnotu, kterou systém dokáže reprezentovat. Rozdíl mezi původní analogovou hodnotou a zaokrouhlenou digitální hodnotou se nazývá kvantizační chyba, která se v signálu projevuje jako kvantizační šum.

Velikost kvantizačního šumu závisí na bitové hloubce. Čím více bitů použijeme, tím více úrovní amplitudy může být zaznamenáno, a tím je kvantizační šum menší. Například:

- 8bitový systém – 2^8 úrovní (256) – vyšší šum
- 16bitový systém – 2^{16} úrovní (65 536)
- 24bitový šum – 2^{24} úrovní (16 777 216) – velmi nízký kvantizační šum

Pro minimalizaci slyšitelnosti kvantizačního šumu se v praxi často používá dithering, tedy přidání nízkoúrovňového šumu před kvantizací. Tím se šum stane náhodnějším a méně slyšitelným, zejména v tichých pasážích.

Aliasing

Aliasing je jev, který vzniká při nedostatečné vzorkovací frekvenci během A/D převodu. Pokud není dodrženo pravidlo, že vzorkovací frekvence musí být alespoň dvojnásobkem nejvyšší frekvence ve vzorkovaném signálu (tzv. Nyquistova frekvence), dochází k tomu, že vyšší frekvence se „přelijí“ do nižších a ve výsledném digitálním signálu se objeví falešné frekvence, které v původním analogovém signálu nebyly.

Tento jev je slyšitelný jako zkreslení a může významně snížit kvalitu nahrávky. Aby se tomu zabránilo, je běžnou praxí použití anti-aliasing filtru – jedná se o dolní propust, která odstraní frekvence nad hranicí vzorkování ještě před samotným digitalizováním signálu.

Dynamický rozsah

Dynamický rozsah digitálního audiosystému definuje rozsah mezi nejhlasitějším a nejtišším možným zaznamenatelným zvukem, který je systémem ještě zachytitelný bez zkreslení. Tento rozsah je přímo závislý na bitové hloubce, přičemž každý bit navíc přidává přibližně 6 dB dynamického rozsahu:

- 16bitový systém (např. CD) má dynamický rozsah cca 96 dB,
- 24bitový systém (běžný v profesionální produkci) má dynamický rozsah cca 144 dB.

Vyšší dynamický rozsah umožňuje přesněji zaznamenat jemné nuance a detaily ve velmi tichých i hlasitých pasážích bez výrazného šumu nebo zkreslení. To je zvlášť důležité při nahrávání akustické hudby, filmové produkce či masteringových úprav. (Stolet, 2009)

1.2.3 Formáty digitálního zvuku

Po digitalizaci zvukového signálu je důležité zvolit vhodný formát pro jeho uložení. Existuje celá řada formátů digitálního zvuku, které se liší způsobem komprese, kvalitou záznamu, velikostí souboru a kompatibilitou s různými zařízeními a softwary. Volba formátu závisí na konkrétním účelu – zatímco nekomprimované formáty se hodí pro archivaci nebo profesionální zpracování, ztrátově komprimované formáty jsou vhodnější pro běžný poslech nebo streamování díky menší datové náročnosti.

Veškeré digitální formáty můžeme rozdělit do čtyř velkých skupin, formáty ztrátové, bezztrátové, komprimované a nekomprimované. Z předchozího textu je zřejmé, že audio soubory při digitalizaci mohou procházet určitou kompresí dat, která záleží na vzorkovací frekvenci a na bitové hloubce procesu. Přehled nejznámějších audio formátů včetně rozdělení a dalších dat či vlastností pro lepší představu je v následující tabulce 2. (Svetlik, 2025; Trivedi, 2016; Gordon, 2012)

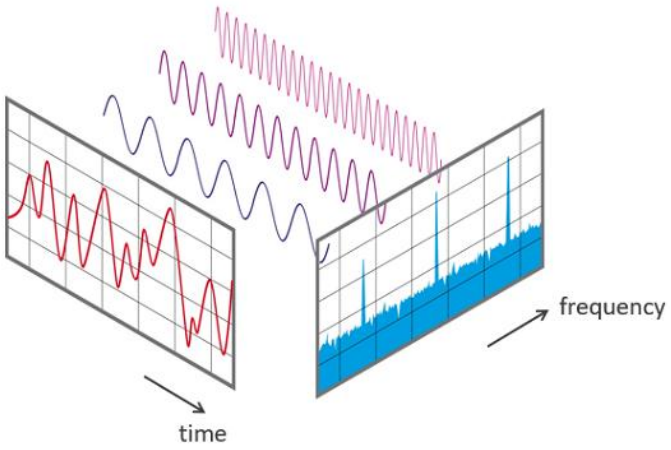
Tabulka 2 - přehled nejpoužívanějších audio formátů (Svetlik, 2025; Trivedi, 2016; Gordon, 2012)

Název	Míra komprese	SR a bitová hloubka/bitrate	Data nahrávky (MB/min.)	Použití
WAV	Bez komprese, bez ztrát	44,1 kHz, 16 bitů	10,0	Prof. prostředí, archivace
AIFF	Bez komprese, bez ztrát	44,1 kHz, 16 bitů	10,0	Prof. prostředí, archivace (Apple)
APE	Velká, bez ztrát	44,1 kHz, 16 bitů	4,0	Méně používaný
MP3	Velká, mnoho ztrát	Bitrate 192 kbps	1,4	Hudba
AAC	Velká, mnoho ztrát	Bitrate 192 kbps	1,4	Hudba (Apple)

1.3 Frekvenční analýza digitálního audia

1.3.1 Úvod do frekvenční analýzy

Lidské vnímání zvuku je do značné míry spojeno s frekvencí – od rozpoznání výšky tónu až po barvu zvuku. Zatímco v časové oblasti nám audio signál ukazuje, jak se mění tlak vzduchu (nebo elektrické napětí) v čase, frekvenční analýza nám umožňuje zkoumat, jaké frekvence jsou v daném signálu přítomny a jak se jejich intenzita v průběhu času mění. To je klíčové nejen pro analýzu hudby, ale také pro různé aplikace ve strojovém učení, detekci řeči, zpracování signálu a transkripci hudby. Skvěle tento princip reprezentuje obrázek 13, na kterém je zobrazeno, jak se jednotlivé frekvenční složky audio signálu projevují a mění.



Obrázek 13 - ukázka projevu signálů s různými frekvencemi v celkovém audio signálu, zachycená v časové a odpovídající frekvenční doméně (NTi Audio, 2025))

Pro tuto analýzu existuje několik matematických nástrojů, které umožňují převod signálu z časové oblasti do frekvenční: Fourierova transformace (FT), rychlá Fourierova transformace (FFT), krátkodobá Fourierova transformace (STFT) nebo konstantním-Q transformace (CQT). Každá z těchto metod má své výhody a nevýhody, které závisí na konkrétním použití a povaze analyzovaného signálu.

1.3.2 Fourierova transformace: spojitá, diskrétní a krátkodobá analýza

Cílem této bakalářské práce je přiblížit čtenáři princip automatizované transkripce hudebních nahrávek do notového zápisu pomocí neuronových sítí, nikoliv detailně vysvětlit principy frekvenčních analýz. Právě proto bude následující text obsahovat zjednodušené vysvětlení principu Fourierovy transformace a jejích typů.

Spojitá Fourierova transformace (FT)

Spojitá FT je definována následovně:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt \qquad (1)$$

kde:
F(ω) je Fourierova transformace f(t),
ω je úhlová frekvence,
i je imaginární číslo,
t je čas.
Fourierova transformace je matematický nástroj definovaný rovnicí (1), který umožňuje rozložit funkci nebo signál na jeho jednotlivé frekvenční složky. Umožňuje převod signálů mezi dvěma oblastmi – konkrétně mezi časovou a frekvenční doménou. Její hlavní výhodou je možnost rozkladu i neperiodických funkcí na jednoduché sinusové složky, čímž se usnadňuje jejich analýza. Rozlišujeme dva základní typy FT: přímou FT a inverzní FT. Pomocí těchto dvou typů můžeme převést data z časové oblasti do frekvenční a poté zpět do časové. Jde o velmi účinný přístup, využívaný napříč obory, jako je zpracování signálů, fyzika či inženýrství, k analýze frekvenčního obsahu proměnlivých signálů nebo funkcí v čase či prostoru.

Výstupem spojité FT je komplexní funkce F(ω), která popisuje, jaké frekvenční složky se nacházejí v původním časovém signálu a s jakou amplitudou a fází. Tato funkce je definována na spojitém intervalu úhlových frekvencí ω (v radiánech za sekundu), a proto má výstup formu spojitého spektra.

Hodnota |F(ω)| reprezentuje amplitudu složky o frekvenci ω, zatímco argument komplexního čísla arg(F(ω)) určuje fázi této složky. Celkově lze tedy říci, že spojitá FT poskytuje frekvenční spektrum vstupního signálu, kde každému bodu ve frekvenční oblasti odpovídá určitá komplexní hodnota. (GeeksforGeeks, 2024)

Diskrétní Fourierova transformace (DFT)

DFT je definována následovně:

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi nm/N} \qquad (2)$$

kde:
X(m) je m-tá hodnota DFT x(n),
m je index hodnoty DFT ve frekvenční oblasti, m = 0, 1, 2, …, N-1,
x(n) je sekvence vstupů z časové oblasti, x(0), x(1), atd.,
n je index hodnot v časové oblasti, n = 0, 1, 2, …, N-1,
N je počet vzorků v časové oblasti a počet frekvenčních bodů ve výstupu DFT,
i je imaginární číslo.
Diskrétní Fourierova transformace definovaná rovnicí (2) představuje jeden z nejpoužívanějších a zároveň nejvýkonnějších nástrojů v oblasti digitálního zpracování signálů. DFT nám umožňuje analyzovat, upravovat a syntetizovat signály způsoby, které v rámci spojitého – tedy analogového – zpracování nejsou možné. I když se dnes uplatňuje téměř ve všech technických oborech, její význam nadále roste, jak se její potenciál stále více rozpoznává.

DFT je matematický postup určený ke zjištění harmonického, tedy frekvenčního, obsahu diskrétní posloupnosti signálu. V našem kontextu je touto posloupností sada hodnot získaných periodickým vzorkováním spojitého signálu v čase. Nicméně ukazuje se, že DFT je užitečná pro analýzu libovolných diskrétních dat – bez ohledu na to, co přesně reprezentují.

Původ DFT vychází ze spojité Fourierovy transformace X(f), která převádí časovou funkci x(t) do jejího frekvenčního obrazu X(f). Tento převod umožňuje zjistit frekvenční složení daného signálu a otevírá široké možnosti pro jeho analýzu a zpracování, zejména v inženýrství a fyzice. S nástupem digitálních počítačů a rozvojem digitální techniky byla postupně vyvinuta DFT, definovaná jako diskrétní frekvenční posloupnost X(m). Pro náš výklad tato transformace vychází z diskrétní časové posloupnosti x(n), vzniklé vzorkováním spojitého signálu x(t). (Lyons, 2011)

Krátkodobá Fourierova transformace (STFT)

STFT je definována následovně:

$$Sf, \tau = \sum_{t=0}^{N-1} x(t)\omega(t) - \tau e^{-i2\pi ft} \quad (3)$$

kde:
x(t) je analyzovaný signál,
ω(t) je okénková funkce,
τ je čas.
Krátkodobá Fourierova transformace definována rovnicí (3) byla zavedena jako řešení určitých nedostatků klasické FFT. Obvykle se používá k analýze úzkopásmových frekvenčních složek v nestacionárních nebo šumem zasažených signálech. Základní myšlenkou STFT je rozdělit původní signál do krátkých časových úseků (oken) a na každý tento segment následně aplikovat Fourierovu transformaci. Tím je možné popsat, jak se frekvenční složení signálu mění v čase.

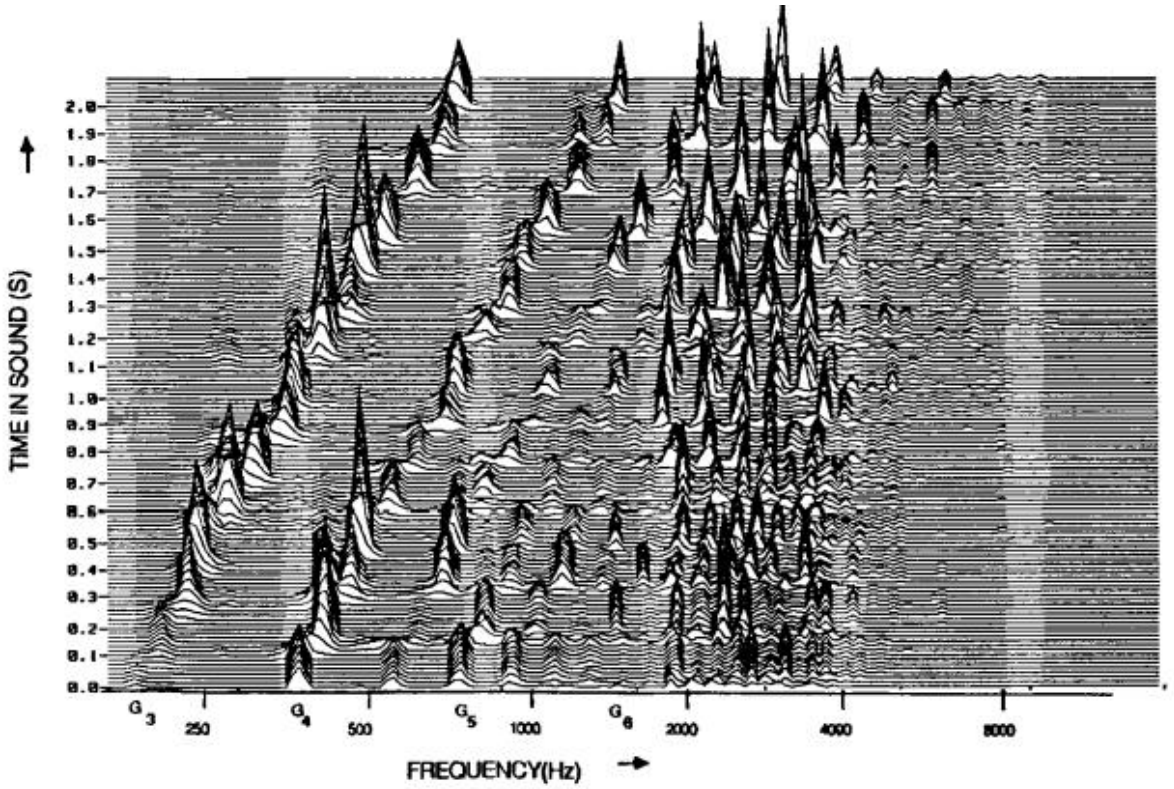
Tato metoda nabízí konstantní šířku pásma v analýze, což umožňuje identifikaci harmonických složek, a zároveň poskytuje konstantní rozlišení v časově-frekvenčním zobrazení – bez ohledu na konkrétní frekvenci. (Goyal a Pabla, 2015)

1.3.3 Konstantní-Q transformace a její harmonická varianta

Konstantní-Q transformace je forma frekvenční analýzy, kterou je nejvhodnější používat na hudební analýzu, protože využívá logaritmické rozdělení frekvenčních pásem. Toto rozdělení odpovídá lidskému sluchu na rozdíl od FT, DFT nebo STFT, a navíc odpovídá tónovému rozdělení klasické temperované stupnice. Frekvenční pásma jsou rozdělena tak, že každé pásmo má stejný poměr mezi střední frekvencí a šířkou pásma (tzv. konstantní Q faktor). To znamená, že nižší frekvence mají vyšší frekvenční rozlišení a delší časová okna, zatímco vyšší frekvence mají nižší frekvenční rozlišení a kratší časová okna. Díky své přirozené afinitě k hudebnímu ladění je CQT ideální pro zpracování hudebních nahrávek, kde tóny nejsou rovnoměrně rozloženy po spektru, ale spíše podle logaritmického vztahu mezi výškami tónů. To umožňuje přesnější identifikaci a separaci jednotlivých not i akordů v polyfonní hudbě.

Harmonická Constant-Q transformace (HCQT) je rozšířením CQT, které zahrnuje analýzu několika harmonických násobků základní frekvence. Typicky se vypočítá několik CQT transformací, z nichž každá je zaměřena na konkrétní násobek základní frekvence (např. 2f, 3f, 4f atd.). Výsledkem je vícerozměrné spektrum (obvykle 3D tenzor), kde každá „vrstva“ odpovídá určitému harmonickému řádu. Tato metoda je zvláště užitečná při analýze hudebních

signálů, kde harmonické struktury hrají klíčovou roli. HCQT vytváří vícerozměrné reprezentace signálu, které zachycují energii v základní frekvenci a jejích harmonických. Tímto způsobem poskytuje bohatší informace o spektrální struktuře signálu, což je cenné například při rozpoznávání hudebních nástrojů nebo při transkripci hudby. (Brown, 1991)



Obrázek 14 - graf CQT analýzy houslí hrajících diatonickou stupnici (Brown, 1991)

1.4 Využití neuronových sítí v hudební transkripci

1.4.1 Vznik neuronových sítí a hlubokého učení

Neuronové sítě, dnes hojně využívané v oblasti umělé inteligence, mají své kořeny již v roce 1944, kdy Warren McCulloch a Walter Pitts navrhli první matematický model umělého neuronu. Jejich práce ukázala, že teoreticky lze sítěmi simulovat libovolné výpočetní operace.

V 60. letech však výzkum neuronových sítí zpomalil po kritice Marvina Minskyho a Seymoura Paperta, kteří poukázali na omezenou schopnost tehdejších jednoduchých sítí – konkrétně perceptronů – řešit složitější problémy.

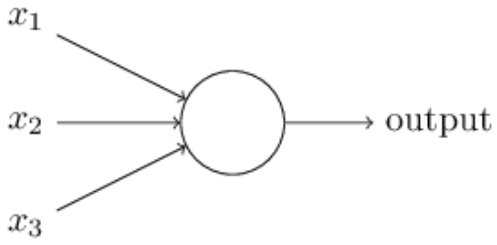
Obnovený zájem nastal v 80. letech díky zavedení vícevstupových sítí a algoritmu zpětné propagace chyb. Přesto byly tyto modely náročné na výpočetní výkon a prakticky obtížně použitelné.

Zásadní průlom nastal až po roce 2010, kdy rychlý vývoj výpočetní techniky – zejména GPU – umožnil efektivní trénování hlubokých neuronových sítí. Tento přístup, označovaný jako deep learning, dnes dominuje v oblastech jako rozpoznávání řeči, obrazu nebo hudby. (Hardesty, 2017)

1.4.2 Co jsou neuronové sítě

Pro přiblížení principu neuronových sítí je vhodné nejprve vysvětlit, co je perceptron, zmíněný v předchozí kapitole.

Perceptrony jsou typy neuronu, které vznikaly v 50. a 60. letech minulého století pod taktovkou Franka Rosenblatta. Princip činnosti perceptronů, který je zobrazen na obrázku 15, je následující: vstupem do perceptronu je několik binárních hodnot (x_1, x_2, x_3, …), každý binární vstup je násoben reálnou váhou (w_1, w_2, w_3, …), výstupem je opět binární hodnota určená porovnáním váženého součtu $\sum_j w_j x_j$ s reálnou prahovou hodnotou. Pokud tuto prahovou hodnotu přesuneme na druhou stranu rovnice, k váženému součtu, mluvíme o tzv. biasu (Nielsen, 2015)



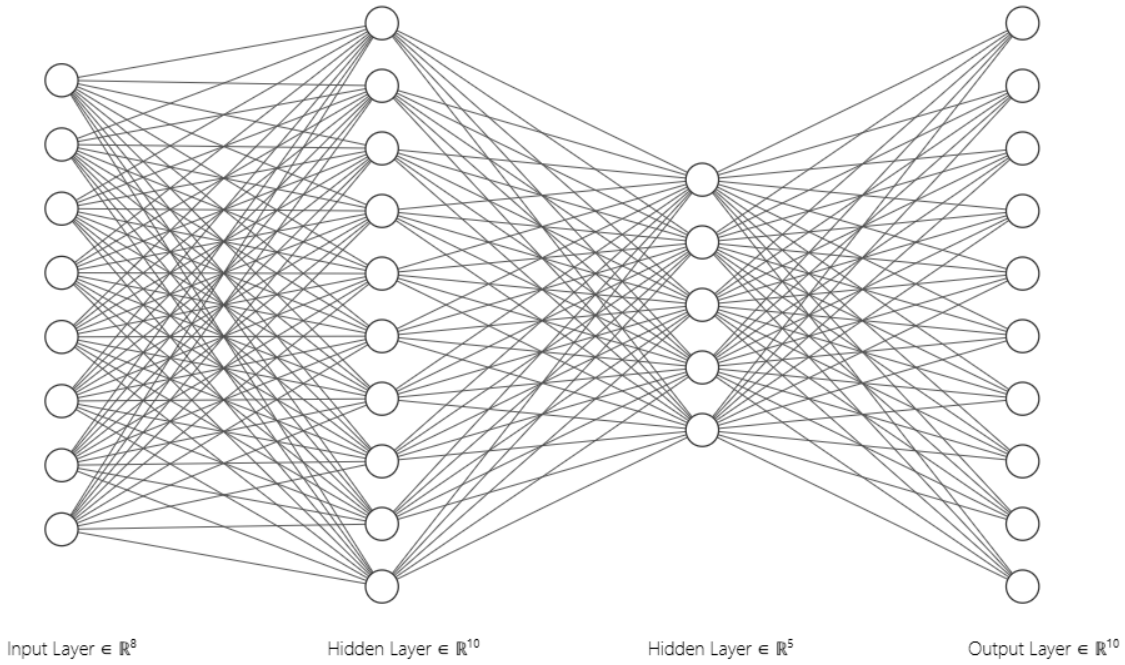
Obrázek 15 - zjednodušené zobrazení principu perceptronu (Nielsen, 2015)

Neurony moderních neuronových sítí fungují na podobném principu jako perceptrony – každý neuron přijímá vstupy, které jsou násobeny příslušnými vahami. Výsledná hodnota se poté zpracuje pomocí vhodné hladké, spojité a diferencovatelné aktivační funkce, o kterých bude řeč v následujících kapitolách. Vstupy, váhy i výstupy jednotlivých neuronů jsou obvykle reprezentovány pomocí vektorů nebo matic, což umožňuje efektivní paralelní výpočty na grafických procesorech (GPU) a výrazně zrychluje trénování i vyhodnocování sítí.

Samotné neuronové sítě (NN) poté vznikají uspořádáním neuronů a spojením jejich výstupů se vstupy jiných (nebo stejných) neuronů. Celá NN se poté dělí do vrstev. Vrstva je soubor neuronů, které zpracovávají informace paralelně. Mezi jednotlivé vrstvy NN patří:

- vstupní vrstva,
- skryté vrstvy – umožňují NN modelovat složitější nelineární vztahy mezi vstupem a výstupem,
- výstupní vrstva.

Na obrázku 16 je znázorněna architektura neuronové sítě s 8 vstupními, 15 skrytými (ve 2 vrstvách) a 10 výstupními neurony. Architektura neuronové sítě popisuje, jak jsou jednotlivé vrstvy a neurony mezi sebou propojeny, jaký je jejich počet a jakým způsobem spolu komunikují.



Obrázek 16 - ukázka architektury NN pomocí online nástroje NN-SVG (LeNail, 2019)

1.4.3 Architektura neuronových sítí a její význam

Architekturu NN můžeme měnit nejen počtem neuronů a vrstev, ale i druhem jednotlivých vrstev. Nejčastěji používané druhy NN budou v této kapitole představeny a budou zmíněny i vrstvy, ze kterých se nejčastěji skládají, a funkce těchto sítí a vrstev.

Vícevrstvý perceptron (multilayer perceptron)

Multilayer perceptron (MLP) je typ dopředné neuronové sítě, kde jsou všechny neurony navzájem plně propojené a využívají nelineární aktivační funkce. Uplatňuje se při řešení problémů, které nejsou lineárně oddělitelné. MLP se široce používá v oblastech, jako je rozpoznávání obrazu, zpracování přirozeného jazyka nebo rozpoznávání řeči. Díky své flexibilní architektuře a schopnosti aproximovat libovolné funkce je základním stavebním kamenem hlubokého učení.

Struktura MLP:

- vstupní vrstva – obsahuje neurony odpovídající jednotlivým rysům vstupních dat,
- skryté vrstvy – zajišťují zpracování signálu; každá neuronová vrstva je plně propojena s předchozí vrstvou a počet vrstev i neuronů je volen při návrhu sítě,
- výstupní vrstva – generuje konečný výstup; počet neuronů závisí na typu úlohy (např. binární nebo více třídní klasifikace),
- váhy – určují sílu spojení mezi neurony a během trénování se upravují, aby se minimalizovala chyba sítě,
- bias neurony – umožňují síti flexibilněji posouvat aktivační funkce a lépe se přizpůsobit trénovacím datům.

Každý neuron aplikuje na svůj výstup aktivační funkci (např. sigmoid, tanh, ReLU, softmax), čímž se do sítě vnáší nelinearita nezbytná pro učení složitějších vztahů. MLP se učí pomocí algoritmu zpětného šíření chyb (backpropagation), který pomocí gradientního sestupu optimalizuje váhy tak, aby minimalizoval chybu modelu. (Jaiswal, 2025)

Konvoluční neuronová síť (convolutional neural network)

Convolutional Neural Networks, zkráceně CNN nebo ConvNets, jsou speciální druhy hlubokých neuronových sítí navržených především pro úlohy rozpoznávání objektů, jako je klasifikace, detekce nebo segmentace obrazů. CNN najdeme v praxi například v autonomních vozidlech nebo bezpečnostních kamerových systémech.

CNN se skládá ze čtyř hlavních bloků:

- Konvoluční vrstvy – umožňují detekci vzorů v obraze pomocí filtrů (kernelů), které se během trénování naučí hledat konkrétní znaky, jako jsou hrany nebo tvary.
- ReLU (rectified linear unit) – po konvoluci se na výstupy aplikuje ReLU aktivační funkce, která vnáší nelinearitu a pomáhá v boji proti problémům s mizícími gradienty.
- Pooling vrstvy – tyto vrstvy zjednodušují a zmenšují velikost obrazu agregací (např. max-poolingem), což pomáhá snížit paměťovou náročnost a snižuje riziko přeučení.
- Plně propojené vrstvy (fully connected layers) – výstup z pooling vrstev je zploštěn a přiváděn do klasických neuronových vrstev, kde se pomocí funkcí jako softmax generují pravděpodobnosti pro jednotlivé třídy.

CNN byly inspirovány fungováním lidského vizuálního systému. Stejně jako mozek mají hierarchickou strukturu, kdy v počátečních vrstvách extrahují jednoduché příznaky a v hlubších vrstvách postupně složitější rysy. Pracují také s lokální konektivitou, podobně jako neurony v mozku reagují pouze na omezenou část zorného pole. Díky pooling vrstvám dosahují translační invariance, což znamená, že dokážou rozpoznat prvky bez ohledu na jejich přesné umístění v obraze. V každé vrstvě vytvářejí několik map příznaků, které zachycují různé typy detekovaných vzorů. Nelineární reakce neuronů v mozku napodobují pomocí aktivačních funkcí, jako je například ReLU. Přestože CNN nejsou tak složité jako lidský mozek a postrádají jeho propracované zpětnovazební mechanismy, i tak znamenaly velký pokrok v oblasti počítačového vidění. (Keita, 2023)

Rekurentní neuronová síť (recurrent neural network)

Rekurentní neuronové sítě (RNN) představují specifický typ umělé neuronové sítě navržený pro zpracování sekvenčních dat, jako jsou časové řady, texty nebo řeč. Jejich klíčovou vlastností je vnitřní paměť, která jim umožňuje uchovávat informace o předchozích vstupech a využívat je při zpracování aktuálních dat. Díky tomu jsou vhodné například pro predikci hodnot na základě časových řad, strojový překlad, rozpoznávání řeči nebo generování textu.

Základní principy RNN jsou následující:

- Sekvenční zpracování dat – RNN přijímá vstupy postupně, krok za krokem (např. slova ve větě nebo hodnoty v časové řadě), a každý nový vstup ovlivňuje nejen aktuální výstup, ale i vnitřní stav sítě; podobnost se sekvenčními logickými obvody.
- Sdílené váhy – ve všech časových krocích používá RNN stejné váhy a biasy, což zjednodušuje učení a zajišťuje konzistentní zpracování sekvencí.
- Učení pomocí BPTT (Backpropagation Through Time) – RNN se učí tak, že šíří chybu zpětně přes všechny časové kroky, což umožňuje optimalizovat váhy na základě celé posloupnosti.

RNN se využívají v různých variantách podle charakteru úlohy – například pro převod textu na kategorii (many-to-one), nebo pro strojový překlad, kde vstupem i výstupem je sekvence (many-to-many). Základní RNN však trpí problémy s tzv. mizejícím nebo explodujícím gradientem, což omezuje jejich schopnost pracovat s dlouhými sekvencemi. Tyto nedostatky řeší pokročilejší architektury, jako jsou LSTM (Long Short-Term Memory) a GRU (Gated Recurrent Unit), které dokážou uchovat informace po delší dobu a efektivněji rozhodovat o tom, co si pamatují a co zapomenou. (Awan, 2022)

1.4.4 Co jsou datové sady

Dataset (neboli datová sada) je uspořádaný soubor dat, která jsou vzájemně propojená a obvykle pocházejí z jednoho zdroje nebo slouží konkrétnímu účelu. Typickým příkladem může být soubor obchodních dat obsahující informace o prodejích, zákaznících, transakcích nebo chování uživatelů. Data v rámci datasetu mohou být číselná, textová, obrazová či zvuková – často se vyskytují i ve vzájemné kombinaci. Důležité je, že jednotlivé záznamy lze zpravidla analyzovat samostatně, ve skupinách nebo jako celek.

Dataset je základním nástrojem pro datovou analýzu, umělou inteligenci a strojové učení. Představuje zdroj informací, ze kterého lze získávat poznatky, sledovat trendy a vytvářet prediktivní modely. Zejména ve strojovém učení hraje výběr správného datasetu klíčovou roli – bez vhodných a kvalitních dat nelze vytvořit efektivní model.

Každý dataset má určitou vnitřní organizaci, která usnadňuje práci s daty a jejich zpracování. Tato struktura obvykle zahrnuje:

- proměnné (variables) – jednotlivé atributy nebo vlastnosti sledované v rámci datasetu, například „cena“, „datum nákupu“ nebo „kategorie zboží“,
- schéma (schema) – rozvržení a vztahy mezi proměnnými, u tabulkových dat definuje např. názvy sloupců a datové typy, u JSON strukturu objektů a polí,
- metadata – informace o samotném datasetu – jeho původ, účel, způsob sběru a doporučení pro použití.

Datasetem tedy není jakákoli nahodilá skupina dat. Aby bylo možné datovou sadu efektivně analyzovat a využít, musí být organizovaná a smysluplně strukturovaná. (Badman a Kosinski, 2025)

1.4.5 Proces učení neuronových sítí

Proces učení neuronových sítí je založen na iterativním upravování vah a biasů (posunů) tak, aby síť postupně zlepšovala své předpovědi. Učení probíhá ve dvou hlavních fázích: forward propagation (dopředné šíření) a backpropagation (zpětné šíření).

Dopředné šíření (forward propagation)

Když do neuronové sítě vstoupí vstupní data, jsou předávána vrstvami sítě – od vstupní vrstvy přes skryté vrstvy až k výstupní. Tato fáze slouží k výpočtu výstupu sítě na základě aktuální konfigurace vah.

Lineární transformace znamená, že každý neuron přijímá vstupy, které násobí příslušnými vahami. Výsledky sečte a přičte k nim tzv. bias, tento výpočet lze zapsat jako v rovnici (4):

$$z = \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_n x_n + b \qquad (4)$$

kde ω jsou váhy, x vstupy a b bias.

Aktivace znamená, že výsledný součet z je pak zpracován tzv. aktivační funkcí, která zavádí do výpočtu nelinearitu, bez ní by síť nebyla schopna modelovat složité vztahy v datech, mezi běžně používané aktivační funkce patří ReLU (Rectified Linear Unit), sigmoid nebo tanh.

Díky této fázi síť získá svůj první odhad výstupu pro daný vstup, ale zatím ještě neví, jak přesný je tento odhad.

Zpětné šíření (backpropagation)

Po dopředném průchodu sítí se výstup porovná s požadovaným výsledkem. Tento rozdíl se kvantifikuje pomocí tzv. ztrátové funkce (loss function). Úkolem neuronové sítě je tuto chybu minimalizovat.

Proces zpětného šíření zahrnuje:

- Výpočet chyby – síť vypočítá rozdíl mezi skutečným výstupem a očekávaným výstupem, v regresních úlohách se často používá střední kvadratická chyba (MSE), v klasifikaci zase například křížová entropie (cross-entropy).
- Výpočet gradientů – pomocí derivací a řetězového pravidla (chain rule) síť určí, v jaké míře každý jednotlivý parametr (váha nebo bias) ovlivnil výslednou chybu, tento krok je klíčový pro následnou úpravu parametrů.

- Aktualizace parametrů – v posledním kroku jsou váhy a biasy upraveny v opačném směru, než ukazují gradienty. Aby se chyba v příští iteraci zmenšila, rychlost těchto změn určuje tzv. learning rate (učicí rychlost).

V některých případech se používají i pokročilejší optimalizační algoritmy jako Adam nebo RMSProp, které dynamicky upravují velikost kroku pro každý parametr zvlášť.

Iterace a konvergence

Tento celý proces – dopředné šíření, výpočet chyby, zpětné šíření a aktualizace vah – se opakuje mnohokrát, často ve stovkách až milionech cyklů (epoch), dokud se síť „nenaučí“ co nejpřesněji předpovídat výstupy pro danou úlohu neboli než dosáhne síť co největší konvergence. Konvergence v kontextu učení neuronových sítí znamená, že se síť postupně přibližuje k optimálnímu řešení – tedy k takovému nastavení vah a biasů, při kterém je ztrátová funkce (loss) minimální, a tím i předpovědi co nejpřesnější. (GeeksforGeeks, 2025)

1.4.6 Metriky přesnosti neuronových sítí

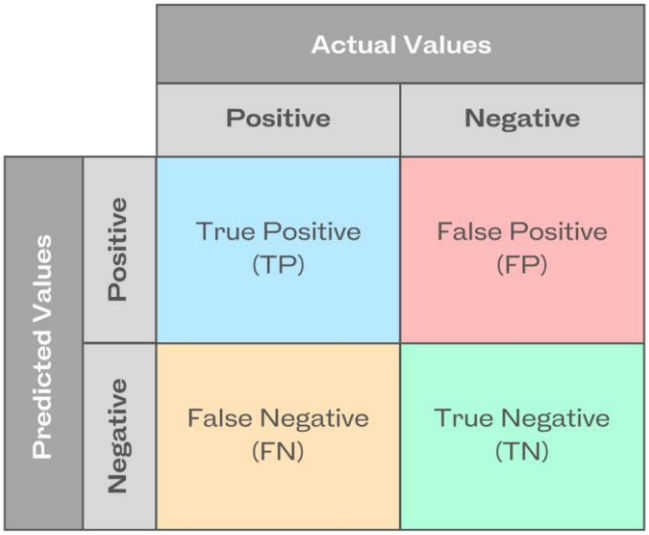
Při trénování neuronových sítí nestačí sledovat pouze hodnotu ztrátové funkce, která vyjadřuje rozdíl mezi predikcí modelu a skutečnými hodnotami. Aby bylo možné objektivně posoudit, jak dobře model funguje, je nezbytné využít specifické metriky přesnosti. Tyto metriky umožňují kvantifikovat výkonnost modelu z různých pohledů, v závislosti na typu úlohy – ať už jde o klasifikaci, regresi nebo detekci sekvencí.

Výběr správné metriky hraje klíčovou roli nejen při hodnocení výsledků, ale také při ladění modelu a rozhodování o jeho praktickém nasazení. V této kapitole jsou uvedeny nejčastěji používané metriky pro různé typy úloh a vysvětlen jejich význam i vhodnost použití v konkrétních scénářích.

Pro pochopení vzorců pro výpočet metrik je nutné pochopit tzv. matici záměn na obrázku 17. Matice záměn (confusion matrix) je základní nástroj pro vyhodnocení výkonu klasifikačního modelu. Umožňuje přesně zjistit, jak si model vede při správném i nesprávném určování tříd.

Obsahuje čtyři základní kategorie:

- True Positive (TP) – model správně označil pozitivní případ jako pozitivní,
- True Negative (TN) – model správně označil negativní případ jako negativní,
- False Positive (FP) – model nesprávně označil negativní případ jako pozitivní (tzv. „falešný poplach“),
- False Negative (FN) – model nesprávně označil pozitivní případ jako negativní (tzv. „přehlédnutí“).



Obrázek 17 - matice záměn (Buhl, 2023)

Přesnost (accuracy)

Přesnost vyjadřuje, jaký podíl všech klasifikací byl správný – ať se jednalo o pozitivní nebo negativní výstupy. Vzorec pro výpočet (5) zahrnuje všechny čtyři kategorie z matice záměn (TP, TN, FP, FN) a počítá se jako součet správně klasifikovaných případů (TP + TN) dělený celkovým počtem všech případů.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

V případě detekce spamu accuracy ukazuje, jaká část všech e-mailů byla správně zařazena – tedy jak spamové, tak i nespamové zprávy.

Dokonalý model by nevytvářel žádné falešně pozitivní ani falešně negativní klasifikace, a tedy by dosáhl accuracy = 1.0 (100 %).

V dobře vyvážených datasetech, kde je přibližně stejný počet vzorků v obou třídách, lze accuracy použít jako základní měřítko kvality modelu. Z tohoto důvodu bývá často výchozí metrikou pro hodnocení modelů, kde není stanoven specifický cíl nebo kritérium.

Avšak v případech s výrazně nevyváženým počtem tříd, nebo tam, kde mají různé chyby (FP vs. FN) odlišné dopady, není accuracy vhodným měřítkem. Například pokud pozitivní třída tvoří jen 1 % dat, model, který bude tvrdit „vše je negativní“, dosáhne 99% přesnosti – přestože je zcela nepoužitelný.

Úplnost (recall)

Úplnost – někdy také označovaná jako míra pravých pozitivních (True Positive Rate, TPR) – vyjadřuje, jak velký podíl ze všech skutečně pozitivních případů dokázal model správně identifikovat jako pozitivní. Matematicky je recall definován vzorcem (6) jako poměr pravých pozitivních (TP) k součtu pravých pozitivních a falešně negativních (FN) případů.

$$Recall = \frac{TP}{TP+FN} \qquad (6)$$

Falešně negativní výstupy znamenají případy, které měly být klasifikovány jako pozitivní, ale model je omylem označil za negativní – proto jsou zahrnuty ve jmenovateli vzorce. Ve scénáři detekce spamu udává recall, kolik procent všech skutečných spamů bylo modelem správně rozpoznáno. Z tohoto důvodu se někdy recall označuje také jako „pravděpodobnost detekce“ – odpovídá na otázku: „Kolik procent ze všech spamových e-mailů tento model zachytí?“

Model s dokonalou schopností detekce by měl nulový počet falešně negativních případů, a tedy hodnotu recall rovnou 1,0, což odpovídá 100% míře detekce. V případech, kdy je dataset výrazně nevyvážený a pozitivní třída je zastoupena jen zřídka, bývá recall vhodnější metrikou než celková přesnost (accuracy).

Preciznost (precision)

Preciznost označuje podíl všech pozitivních předpovědí modelu, které byly skutečně správné. Jinými slovy, jde o to, kolik z označených pozitivních případů bylo ve skutečnosti opravdu

pozitivních. Z matematického hlediska se preciznost počítá vzorcem (7) jako poměr pravých pozitivních (TP) vůči součtu pravých pozitivních a falešně pozitivních (FP) výsledků.

$$Precision = \frac{TP}{TP+FP} \quad (7)$$

Model, který by nedělal žádné falešně pozitivní chyby (tedy žádný legitimní e-mail by nebyl označen jako spam), by dosáhl ideální preciznosti 1,0. V situacích, kde se pracuje s nevyváženými daty a skutečných pozitivních případů je velmi málo (např. jen 1–2 příklady), má samotná přesnost omezenou vypovídací hodnotu.

Zlepšení preciznosti je dosaženo snížením počtu falešně pozitivních klasifikací. Naproti tomu recall (úplnost) roste, když klesá počet falešně negativních případů. Tyto dvě metriky však často stojí v určitém napětí: například zvýšení prahové hodnoty pro rozhodnutí modelu obvykle snižuje počet falešně pozitivních výsledků (a tím zvyšuje přesnost), ale zároveň zvyšuje počet falešně negativních případů (a tím snižuje úplnost). Výsledkem je, že mezi precizností a úplností bývá často nepřímý vztah – zlepšení jedné může vést ke zhoršení druhé. (Google Developers, 2025)

F1 skóre

F1 skóre (nebo také F-míra) představuje harmonický průměr dvou základních metrik klasifikačního modelu – precision (přesnosti pozitivních predikcí) a recall (úspěšnosti zachycení skutečných pozitivních případů). Obě složky mají v tomto výpočtu (8) stejnou váhu, což zajišťuje, že výsledné skóre věrně odráží schopnost modelu správně identifikovat pozitivní případy.

$$F1\ score = 2 * \frac{precison*recall}{precision+recall} \qquad (8)$$

Hodnoty F1 skóre se pohybují v rozsahu od 0 do 1:

- 0 znamená nejhorší možný výsledek (naprosté selhání modelu),
- 1 značí dokonalý model, který bezchybně klasifikuje všechny případy.

Vysoké F1 skóre indikuje dobře vyvážený model, který dokáže současně dosahovat vysoké přesnosti i senzitivity. Naopak nízké F1 skóre často znamená, že model upřednostňuje jednu metriku na úkor druhé, čímž ztrácí celkovou spolehlivost. (Buhl, 2023)

1.4.7 Nejčastější problémy při práci s neuronovými sítěmi

Ačkoli neuronové sítě dosahují výborných výsledků v celé řadě úloh, jejich úspěch závisí na správném nastavení, kvalitě dat a vhodné metodologii. Při trénování modelu se často objevují typické problémy, které mohou výrazně ovlivnit výkonnost výsledného systému. V této části jsou popsány nejčastější z nich, včetně přeučení (overfitting), nedotrénování (underfitting), práce s nevyváženými daty a dopad nedostatečného předzpracování dat.

Přeučení (overfitting)

Přeučení nastává tehdy, když se model naučí velmi dobře rozpoznávat vzory v trénovacích datech, ale jeho schopnost zobecnit znalosti na nová, dosud neviděná data je slabá. K tomuto jevu často dochází v situacích, kdy má model příliš mnoho parametrů, je trénován příliš dlouho, nebo pokud je objem trénovacích dat nedostatečný. Mezi možná řešení patří:

- Dropout – náhodné vypínání některých neuronů během tréninku nutí model nespoléhat se příliš na konkrétní části sítě a zvyšuje tak jeho odolnost vůči přeučení.
- Předčasné zastavení (early stopping) – trénink se ukončí v okamžiku, kdy se validační ztráta po určitou dobu nezlepšuje, tím se zamezí, aby se model přizpůsoboval šumu ve trénovacích datech.
- Rozšíření dat (data augmentation) – umělé zvětšení množiny trénovacích dat pomocí různých transformací (např. otáčení, změna velikosti, zrcadlení) pomáhá modelu učit se obecnější vzory.

Nedotrénování (underfitting)

K nedotrénování dochází tehdy, když model nedokáže zachytit základní strukturu dat, což vede ke špatnému výkonu jak na trénovacích, tak i validačních datech. Tento problém často vzniká u příliš jednoduchých modelů nebo při nedostatečně dlouhém trénování. Mezi možná řešení patří:

- Zvýšení složitosti modelu – přidáním dalších vrstev nebo neuronů lze modelu umožnit naučit se složitější vzory.
- Delší trénink – navýšením počtu epoch může model lépe porozumět datům a zlepšit své predikce.

- Změna optimalizační metody – vyzkoušení různých optimalizačních algoritmů, jako je Adam, RMSprop nebo Adagrad, může pomoci dosáhnout lepší konvergence a efektivnějšího učení.

Nedostatečná nebo nevyvážená data

Modely neuronových sítí jsou velmi citlivé na kvalitu a množství vstupních dat. Pokud je dataset příliš malý nebo obsahuje výrazně nerovnoměrné zastoupení jednotlivých tříd, může to vést k nepřesným výsledkům a špatné schopnosti modelu generalizovat. Mezi možná řešení patří:

- Rozšíření dat (data augmentation) – pomocí různých transformací (např. otáčení, škálování, zrcadlení) lze z existujících dat vytvořit nové příklady, čímž se efektivně zvětší trénovací množina.
- Vyvážení dat – lze použít techniky oversamplingu (zvětšení počtu příkladů minoritní třídy) nebo undersamplingu (zmenšení množství dat majoritní třídy), aby došlo k rovnoměrnějšímu zastoupení jednotlivých kategorií.
- Transfer learning – využití již natrénovaných modelů na podobných úlohách umožňuje přenést znalosti do nového úkolu a zlepšit výsledky i při menším množství dostupných dat.

(Lost and Found, 2023)

2 PRAKTICKÁ ČÁST

2.1 Návrh automatického systému pro transkripci hudby

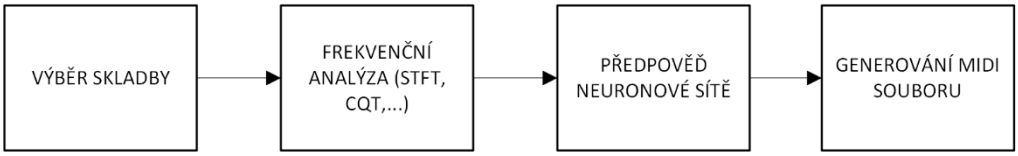
2.1.1 Cíl systému

Cílem navrhovaného systému je automaticky převést audiozáznam hudební skladby do podoby digitálního notového zápisu pomocí formátu MIDI. Hlavním úkolem je detekovat výšku tónů (pitch), jejich časové umístění (onset a offset) a převést tyto informace do struktury odpovídající notovému zápisu.

Systém je navržen tak, aby byl plně automatizovaný a využíval nástroje z oblasti strojového učení, konkrétně hluboké neuronové sítě. Jeho vstupem je zvuková nahrávka hudby v digitálním formátu (např. WAV nebo MP3), výstupem pak generovaný soubor ve formátu MIDI, který obsahuje odpovídající sekvenci not.

Výstupní MIDI lze následně dále zpracovávat v notových editorech nebo použít k analýze hudby, přehrávání syntetizátorem či počítačem nebo k tvorbě partitury.

2.1.2 Návrh systému



Obrázek 18 - blokové schéma návrhu systému (autor, 2025)

Na obrázku 18 je uveden koncept návrhu systému automatické transkripce. Podle tohoto obrázku se systém skládá z několika dílčích procesů:

- výběr skladby k transkripci
- extrakce frekvenčních vlastností hudebního signálu pomocí transformací (CQT, STFT)
- průchod extrahovaných dat neuronovou sítí a předpověď not (onset, offset a pitch)
- generování MIDI souboru

Systém je zaměřen především na jednohlasé nebo slabě vícehlasé skladby, a to s cílem dosažení co nejvyšší přesnosti detekce v běžných nahrávkách bez potřeby notového vstupu.

2.2 Použité nástroje a knihovny

2.2.1 Programovací jazyk a vývojové prostředí

Pro návrh a implementaci automatického systému pro transkripci hudby byl zvolen programovací jazyk Python ve verzi 3.12. Tento jazyk je v současnosti jedním z nejrozšířenějších nástrojů v oblasti strojového učení a zpracování zvuku, a to zejména díky široké dostupnosti specializovaných knihoven, přehledné syntaxi a silné komunitní podpoře.

Jako hlavní vývojové prostředí byl použit PyCharm ve verzi 2024.2.3 od společnosti JetBrains. Toto integrované vývojové prostředí (IDE) poskytuje pokročilé funkce pro psaní, správu a ladění kódu, což výrazně usnadnilo vývoj a správu projektu. Velkou výhodou tohoto prostředí je také možnost mít pro každý projekt specializovaný Python Interpreter s vlastní sadou knihoven, což je ideální pro alfa testování zvolených přístupů k systému automatické transkripce.

2.2.2 Použité knihovny, frameworky a ostatní aplikace

NumPy (v1.26.4)

Základem pro vědecké výpočty v jazyce Python je knihovna NumPy (Numerical Python). Poskytuje výkonnou podporu pro práci s vícerozměrnými poli (tzv. ndarray), které jsou podstatou práce neuronových sítí, umožňuje efektivní operace nad těmito strukturami a obsahuje množství funkcí pro lineární algebru, statistiku a práci s Fourierovou transformací. NumPy je často využívána jako základ pro další knihovny v oblasti strojového učení, jako jsou TensorFlow nebo PyTorch, které na ní staví své výpočetní jádro.

V rámci této práce je knihovna NumPy využívána pro:

- předzpracování a načtení vstupních dat (např. změna tvaru matic, výběr částí signálu, operace s maticemi),
- efektivní práci s časově-frekvenčními reprezentacemi (např. převod a manipulace s CQT a STFT výstupy),
- přípravu trénovacích a testovacích datasetů.

Librosa (v0.10.2.post)

Librosa je open-source knihovna pro jazyk Python, určená pro analýzu a zpracování hudebních a zvukových signálů. Je široce používána v oblasti hudební informatiky a strojového učení se zvukem. Nabízí sadu nástrojů pro extrakci hudebních příznaků (např. tempo, výška tónu, chromatická analýza), výpočet spektrálních transformací (STFT, CQT, mel-spektrogram atd.), převody mezi frekvencí a MIDI notací, stejně jako základní nástroje pro načítání, zpracování a vizualizaci zvukových dat.

V této práci je Librosa využívána pro:

- výpočet CQT a STFT reprezentace signálu, která slouží jako vstupní data pro neuronovou síť,
- převod hodnot MIDI not na frekvenci v Hz,
- převod hodnot STFT z lineárně rozložené frekvenční osy na logaritmickou osu odpovídající frekvencím 88 kláves klavíru,
- převod hodnot amplitud STFT a CQT na hodnoty v dB.

MIDIUtil (v1.2.1)

MIDIUtil je jednoduchá knihovna pro jazyk Python, která umožňuje programově vytvářet MIDI soubory bez nutnosti ručního zápisu binárního formátu. Uživatel může snadno definovat stopy, noty, délky, tempa nebo nástroje, přičemž všechny požadované parametry lze zadat přímo v kódu. Díky své jednoduchosti a přehlednému rozhraní se MIDIUtil často využívá v projektech, které generují hudební obsah algoritmicky nebo prostřednictvím strojového učení.

V této práci je MIDIUtil využívána pro převod výstupních predikcí neuronové sítě do formátu .mid na základě onsetu, offsetu a MIDI čísla získaného z binární matice.

TensorFlow (v2.18.0) a Keras (v3.6.0)

TensorFlow je populární open-source knihovna vyvinutá společností Google, určená pro numerické výpočty a strojové učení. V oblasti hlubokého učení nabízí bohaté nástroje pro návrh, trénování i ladění neuronových sítí. Umožňuje provoz na různých platformách – od CPU přes GPU až po TPU – a podporuje přechod na větší datové sady i modely. Od verze 2.0 je jeho nedílnou součástí také Keras, které slouží jako jeho high-level API (programové rozhraní aplikace, application programming interface).

Keras zjednodušuje definici a trénování modelů pomocí přehledné objektově orientované syntaxe. Uživatel může snadno sestavit neuronovou síť z předpřipravených vrstev (např. Dense, Dropout, Conv2D, Flatten), zvolit vhodnou ztrátovou funkci, optimalizační algoritmus a metriky, a následně model natrénovat s použitím metody fit().

V této práci jsou knihovny TensorFlow a Keras využívány pro:

- návrh architektury a sestavení neuronové sítě pomocí zjednodušeného objektově orientovaného přístupu pomocí metod tensorflow.keras,
- kompilaci modelu NN pomocí model.compile(),
- trénink modelu NN pomocí model.fit(),
- evaluaci hodnot přesnosti (accuracy) a loss funkce,
- predikci výstupu na neviděných datech.

MuseScore Studio 4 (v4.5.1)

MuseScore Studio 4 je bezplatný open-source program pro sazbu a přehrávání notového zápisu. Nabízí rozsáhlé možnosti tvorby partitury, podporuje import i export ve formátu MIDI, MusicXML, a dalších běžně používaných hudebních formátech. Uživatelé mohou zapisovat noty ručně, nebo importovat hudbu vytvořenou v jiných nástrojích a následně ji upravovat či analyzovat.

V této práci je program MuseScore Studio 4 využit pro kontrolu správnosti výstupu neuronové sítě a závěrečnou úpravu notového zápisu.

Scikit-learn (v1.5.2)

Knihovna Scikit-learn (často označována jako sklearn) je jednou z nejrozšířenějších Python knihoven pro strojové učení. Nabízí široký výběr algoritmů pro klasifikaci, regresi, shlukování, redukci dimenze a další úlohy.

V rámci této práce byla knihovna Scikit-learn využita především pro vyhodnocení úspěšnosti modelu pomocí metrik precision, recall a F1 score.

2.3 Zpracování dat pro učení automatického systému

2.3.1 Kritéria výběru hudebního datasetu

Pro trénování a testování neuronové sítě bylo klíčové zvolit vhodný dataset, který by umožnil efektivní učení modelu a zároveň poskytl dostatečně kvalitní vzorky pro obecnou transkripci hudby. Při výběru datasetu byla zohledněna následující kritéria:

- Dostupnost přesného označení not – dataset musí obsahovat informace o přesném průběhu not, jejich výšce, začátku a délce. Tyto údaje jsou nezbytné pro trénování s dohledem a pro hodnocení přesnosti výsledků.
- Rozmanitost a velikost datasetu – dataset by měl obsahovat dostatečný počet skladeb, ideálně s různými rytmickými i harmonickými strukturami, aby model získal schopnost generalizace.
- Kvalita nahrávek – použitý audio záznam musí být čistý, s minimem šumu a bez výrazných artefaktů, které by mohly negativně ovlivnit frekvenční analýzu.
- Srozumitelný a konzistentní datový formát – dataset by měl být strukturován tak, aby s ním bylo možné efektivně pracovat v prostředí jazyka Python. Ideální je, pokud lze data snadno načíst a převést do podoby běžně používaných struktur, jako jsou např. numpy pole nebo pandas tabulky. To výrazně usnadňuje předzpracování, vizualizaci i vstup do neuronové sítě.

2.3.2 Dataset MusicNet a jeho struktura

MusicNet je veřejně dostupný dataset určený pro úlohy strojového učení v oblasti hudby, zejména pro trénování modelů zaměřených na rozpoznávání not, nástrojů a dalších hudebních struktur. Byl představen v roce 2017 jako první rozsáhlá kolekce klasických hudebních nahrávek propojených s notovým zápisem v přesné časové synchronizaci. Dataset obsahuje více než 30 hodin klasické hudby od skladatelů jako jsou Beethoven, Mozart, Brahms nebo Schubert, interpretovaných na flétnu, klavír, housle, a to jak v sólovém provedení, tak i v duetech, triích či oktetech.

Dataset MusicNet obsahuje více než 330 klasických skladeb různých autorů, přičemž každá skladba je uložena jako dvojice: zvuková nahrávka ve formátu WAV a časově zarovnané anotace ve formě textového souboru. Ke každé skladbě je navíc přiřazeno jedinečné identifikační číslo (ID), které slouží jako klíč pro přístup k datům.

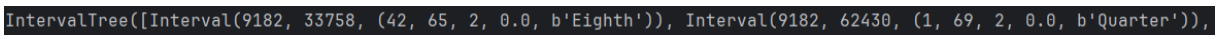
Ukázka 1 – kód pro načítání datasetu MusicNet ze souboru .npz (autor, 2025)

```
data = np.load('musicnet.npz', allow_pickle=True, encoding='bytes')
```

Po načtení dat z oficiálního .npz souboru pomocí knihovny Numpy pomocí kódu v ukázce 1 je výsledkem struktura podobná dvouřádkové matici (přesněji slovníku nebo array) indexované podle ID, které pro každou skladbu obsahuje:

- Na indexu 0 je jednorozměrné pole (vektor) s amplitudami zvukového signálu. Jde o surová data zvukové nahrávky dané skladby, vzorkovaná na 44,1 kHz. Tento signál se dále používá jako vstupní veličina pro převod na frekvenční doménu (pomocí CQT nebo STFT).
- Na indexu 1 se nachází také jednorozměrné pole s objekty IntervalTree, které uchovávají anotace jednotlivých tónů stejným způsobem jako je na obrázku 19. Každý interval v jednotlivých IntervalTree obsahuje nejdůležitější informace o notách v nahrávkách:
    - interval.begin – čas začátku noty (v jednotkách SR*sekundy)
    - interval.end – čas konce noty (v jednotkách SR*sekundy)
    - interval.data – nástroj hrající notu (podle MIDI standardu), znějící nota (podle MIDI standardu), číslo taktu a další data

    Tento intervalový strom umožňuje rychlé dotazování na to, jaké tóny zaznívají v určitém časovém bodě, což je výhodné při generování cílové výstupní matice pro trénink modelu.



Obrázek 19 - ukázka struktury IntervalTree pro skladbu ID 1727 (autor, 2025)

2.3.3 Příprava vstupních dat pro model NN

Z důvodu zpracování velkého množství dat (např. při tréninku na 30 různých skladbách datasetu MusicNet) je celý proces předzpracování vstupních dat zapouzdřen do samostatné user-defined (uživatelem vytvořené) funkce `load_and_process`. Tato funkce má jako vstup pouze identifikátor skladby (`track_id`), na jehož základě dojde k načtení příslušného zvukového signálu a k jeho zpracování funkcí.

Ukázka 2 - definice funkce, načtení amplitud a labels do proměnných, definice parametrů frekvenčních analýz (autor, 2025)

```
def load_and_process(track_id):
    y = data[track_id][0]
    labels = data[track_id][1]

    hop_length = 512
    sr = 44100
    bins_per_octave = 12
    n_bins = 88
    frame_length = np.round(hop_length / sr, 5)
```

Prvním krokem, jak lze vyčíst z ukázky 2, je načtení zvukového signálu a odpovídajících anotací (tedy informací o jednotlivých tónech, labels) ze struktury datasetu uložené v proměnné `data`. Proměnná `y` tedy obsahuje samotný audio signál nahrávek jako pole amplitud, zatímco `labels` reprezentuje anotace pomocích časových intervalů tónů v dané skladbě. Zároveň jsou zde definovány důležité parametry pro následnou frekvenční analýzu, které budou vysvětleny přímo v kontextu s daným typem analýzy, a také výpočet délky snímku (`frame_length`) pomocí velikosti skoku (`hop_length`) a `sr`. Hodnota délky snímku je zaokrouhlena na 5 desetinných míst pomocí `np.round(…, 5)`.

Dalším krokem v rámci předzpracování dat je převod časového signálu do frekvenční oblasti pomocí kódu v ukázce 3 za použití dvou různých transformací – CQT a STFT. Každá z těchto transformací slouží k jinému účelu a umožňuje extrakci různých aspektů z audiosignálu.

Ukázka 3 - kód CQT a STFT analýzy, vytvoření „piano-rollu“ STFT (autor, 2025)

```
cqt = np.abs(librosa.cqt(y, sr=sr, hop_length=hop_length,
bins_per_octave=bins_per_octave, n_bins=n_bins))


n_fft = 4096
stft = np.abs(librosa.stft(y, n_fft=n_fft,
hop_length=hop_length))

piano_freq = librosa.midi_to_hz(np.arange(21, 109))
piano_stft = np.zeros((n_bins, stft.shape[1]))

freq_bins = librosa.fft_frequencies(sr=sr, n_fft=n_fft)
for i, freq in enumerate(piano_freq):
    idx = np.argmin(np.abs(freq_bins - freq))
    start_idx = max(0, idx - 2)
    end_idx = min(len(freq_bins) - 1, idx + 3)
    piano_stft[i] = np.sum(stft[start_idx:end_idx, :], axis=0)
```

Nejprve je výpočtem CQT (`librosa.cqt`) získána vstupní frekvenční reprezentace signálu. Ta rozděluje frekvenční osu na logaritmicky odstupňované úseky, které odpovídají rozložení výšek not. V rámci tohoto kódu je určeno několik parametrů této metody, mezi nimiž jsou:

- `y` – jednorozměrné pole vstupních dat (časová reprezentace zvukové stopy)
- `sr` – vzorkovací frekvence, Hz
- `hop_length` – počet vzorků mezi sousedními CQT okny, horizontální rozlišení CQT
- `bins_per_octave` – počet frekvenčních binů v jedné oktávě
- `n_bins` – celkový počet frekvenčních binů v CQT

Výstupem je matice, kde každý sloupec odpovídá určitému časovému rámci a každý řádek odpovídá konkrétní výšce tónu. Tato matice bude dále v kódu transponována a použita jako část vstupní vrstva NN.

Následně je pomocí STFT (`librosa.stft`) vypočtena lineárně rozdělená frekvenční reprezentace vstupních amplitud, která bude po úpravách a transpozici sloužit jako část vstupní vrstvy NN. Mimo parametru `y` a `hop_length` má STFT ještě upraven parametr `n_fft`, který určuje velikost STFT okna v počtu vzorků, na které se STFT aplikuje.

Pro přiřazení konkrétních tónů byly vytvořeny frekvence odpovídající MIDI tónům od čísla 21 (A_2) po 108 (c^5) pomocí funkce `librosa.midi_to_hz`. Pro každý tento tón je poté v rámci STFT matice vyhledán nejbližší frekvenční bin a v jeho okolí jsou hodnoty sečteny (v rozsahu ±2 biny) pro kompenzaci šířky pásma jednotlivých tónů. Takto vzniklá matice `piano_stft` má stejný časový rozsah jako vstupní CQT reprezentace, obsahuje informace o tom, jak silně byl daný tón přítomen v každém časovém rámci a bude další částí vstupní vrstvy NN.

Pro zvýšení robustnosti vstupních dat a lepší zohlednění hudební harmonie byl na základní CQT signál aplikován jednoduchý model harmonického rozšíření. Tento krok simuluje přirozený jev, kdy se spolu se základním tónem vyskytují i jeho vyšší harmonie, a to zejména v akustických nástrojích.

Ukázka 4 - kód vytvoření matice harmonické CQT analýzy (autor, 2025)

```
harmonic_cqt = np.zeros_like(cqt)
for i in range(n_bins):
    base_idx = i
    harmonic_cqt[base_idx] = cqt[base_idx]
    if base_idx + 12 < n_bins:
        harmonic_cqt[base_idx] += cqt[base_idx + 12]*0.5
    if base_idx + 19 < n_bins:
        harmonic_cqt[base_idx] += cqt[base_idx + 19]*0.3
```

Každý řádek CQT matice, který odpovídá určitému tónu, je rozšířen o přítomnost vybraných vyšších harmonií – konkrétně oktávy (12 půltónů výše) a duodecimy (19 půltónů výše). Tyto vyšší frekvence jsou přičteny k základnímu tónu (pokud jsou v CQT matici) s nižší váhou (0,5 a 0,3), aby reprezentovaly jejich slabší intenzitu v porovnání s tónem základním.

Výsledkem je matice `harmonic_cqt`, která v sobě nese nejen informaci o základních tónech, ale také o jejich harmonickém kontextu. Tato matice je následně použita jako poslední část vstupní reprezentace pro neuronovou síť, která tak získává více hudebního kontextu i v případech, kdy některé základní tóny nejsou v originální nahrávce dominantně slyšitelné.

Pro potřeby trénování neuronové sítě je důležité normalizovat vstupní data do jednotného rozsahu pomocí kódu v ukázce 5. Nejprve je každá z matic (`cqt`, `piano_stft` a `harmonic_cqt`) převedena z amplitudových hodnot na hodnoty v decibelech pomocí

funkce `librosa.amplitude_to_db`, která reflektuje logaritmické vnímání hlasitosti lidským uchem.

Ukázka 5 - kód normalizace vstupních matic (autor, 2025)

```
cqt_norm = librosa.amplitude_to_db(cqt, ref=np.max)
cqt_norm = (cqt_norm - np.min(cqt_norm)) / (np.max(cqt_norm) -
np.min(cqt_norm))

piano_stft_norm = librosa.amplitude_to_db(piano_stft,
ref=np.max)
piano_stft_norm = (piano_stft_norm - np.min(piano_stft_norm))
/ (np.max(piano_stft_norm) - np.min(piano_stft_norm))

harmonic_cqt_norm = librosa.amplitude_to_db(harmonic_cqt,
ref=np.max)
harmonic_cqt_norm = (harmonic_cqt_norm -
np.min(harmonic_cqt_norm)) / (np.max(harmonic_cqt_norm) -
np.min(harmonic_cqt_norm))

input_data = np.concatenate([cqt_norm.T, piano_stft_norm.T,
harmonic_cqt_norm.T], axis = 1)
```

Následně jsou tyto hodnoty převedeny do intervalu $\langle 0, 1 \rangle$ lineární normalizací, čímž se eliminuje vliv absolutní hlasitosti nahrávky a zvýší se stabilita trénovacího procesu.

V závěrečném kroku jsou tyto tři spektrální reprezentace transponovány (tak, aby jednotlivé časové rámce tvořily řádky) a spojeny horizontálně do jediné vstupní matice `input_data`. Tato matice tak v každém řádku obsahuje spojení informací ze základní CQT, STFT aproximované na klaviaturu a harmonicky rozšířené CQT, které neuronová síť dále využívá jako vstupní data.

2.3.4 Příprava výstupních dat pro model NN

Pro trénování neuronové sítě je nezbytné připravit výstupní data, která budou sloužit jako tzv. ground truth – tedy správné odpovědi, na kterých se síť učí. V případě této práce má výstup podobu binární matice, kde řádky reprezentují jednotlivé časové rámce (shodně s maticí vstupních dat) a sloupce odpovídají jednotlivým tónům v rozsahu 88 kláves klavíru.

Každý prvek matice nabývá hodnoty 1, pokud daný tón zněl v odpovídajícím časovém rámci, a hodnoty 0 v opačném případě. Tato binární reprezentace je vhodná pro víceznačkové klasifikační úlohy, kde v jednom čase může znít více tónů současně (například v akordech).

Ukázka 6 - kód pro vytvoření numpy array obsahující onset, offset a MIDI číslo not v nahrávce (autor, 2025)

```
labels_np = np.array([(a.begin, a.end, a.data[1]) for a in
sorted(labels)], dtype=float)
labels_np = labels_np[np.argsort(labels_np[:, 0])]
labels_np[:, :2] = labels_np[:, :2] / sr
labels_np[:, :2] = np.round(labels_np[:, :2] * sr /
hop_length) * hop_length / sr
labels_np[:, :2] = np.round(labels_np[:, :2], 5)
for row in range(labels_np.shape[0]):
    labels_np[row, 0] = np.round(labels_np[row, 0] /
frame_length)
    labels_np[row, 1] = np.round(labels_np[row, 1] /
frame_length)
labels_np = labels_np.astype(int)
```

Část kódu v ukázce 6 se zaměřuje na přípravu anotovaných dat (označení tónů v čase) do struktury vhodné pro učení neuronové sítě. Nejprve se z objektů typu IntervalTree, které obsahují informace o začátku, konci a výšce jednotlivých tónů, vytvoří pole `labels_np`, kde každý řádek reprezentuje jeden tón pomocí trojice: onset (začátek tónu), offset (konec tónu) a jeho výška (ve formátu MIDI čísla).

Následuje několik kroků pro zajištění konzistentního časového zarovnání. Časové údaje jsou převedeny ze vzorků na sekundy a poté opět upraveny tak, aby odpovídaly časovému rozlišení použitém při výpočtu frekvenčních transformací (danému parametrem `hop_length`). Tyto časy jsou zaokrouhleny na hodnoty odpovídající přesným rámcům (frames), a nakonec přepočítány tak, aby místo absolutních časových hodnot obsahovaly indexy příslušných rámců.

Výsledkem je pole `labels_np`, které obsahuje přesně lokalizované a rámcově zarovnané informace o každé notě – tedy vstupní anotace ve formátu vhodném pro vytvoření výstupní trénovací matice.

V závěrečné části funkce v ukázce 7 je vytvořena výstupní binární matice `output_data`, jejímž účelem je sloužit jako cílová (ground-truth) data při trénování neuronové sítě. Tato matice má stejný tvar jako vstupní data `cqt_norm.T` – tedy počet řádků odpovídá časovým rámcům a počet sloupců odpovídá 88 klávesám klavíru (notám v rozsahu MIDI 21–108).

Ukázka 7 - kód vytvářející binární matici hraných not ve skladbě jako výstupní data NN (autor, 2025)

```
output_data = np.zeros(cqt_norm.T.shape, dtype = int)
frame_indices = np.arange(output_data.shape[0])
for start, end, midi_pitch in labels_np:
    note_indices = (frame_indices >= start) & (frame_indices
<= end)
    output_data[note_indices, midi_pitch - 24] = 1

return input_data, output_data
```

Pomocí cyklu se pro každou notu z normalizovaného seznamu anotací `labels_np` určí časové rozmezí jejího trvání (start, end) a příslušné číslo MIDI noty. Pro každý časový rámec v tomto rozmezí je v odpovídajícím řádku matice output_data nastaven sloupec odpovídající dané notě (upravený o posun −24, protože MIDI 21 je v matici na indexu 0) na hodnotu 1. Ostatní hodnoty zůstávají nulové.

Tímto způsobem vzniká tzv. piano roll reprezentace, ve které každá jednička reprezentuje přítomnost konkrétní noty v konkrétním časovém rámci, a síť se tak učí detekovat skutečné noty, které zazněly v podobě tónů v určitých částech skladby.

2.3.5 Příprava trénovacích a testovacích dat

Před samotným trénováním neuronové sítě je nezbytné připravit data ve formátu, který umožní efektivní učení i validaci modelu. Tato fáze zahrnuje rozdělení dat na trénovací a testovací sadu. Níže je popsán způsob, jakým byly jednotlivé vybrané skladby z datasetu MusicNet spojeny do větších celků vhodných pro dávkové zpracování.

Ukázka 8 - výběr tréninkových a testovacích skladeb (autor, 2025)

```
tracks_train = ['2203', '2204', '1733', '1734', '1735',
'2186', '2191', '2217', '2218',
                '2318', '2319', '2320', '2330', '2334',
'2335', '2336', '2570', '2571',
                '2241', '2242', '2243', '2244', '2289',
'2300', '2302', '2303', '2304',
                '2293', '2294', '2295', '2296', '2297']
tracks_test = ['2202']
```

V rámci přípravy dat byla ručně zvolena sada skladeb z datasetu MusicNet, které byly rozděleny do trénovací a testovací množiny, jak je zobrazeno v ukázce 8. Trénovací množina obsahuje celkem 32 skladeb různých autorů a žánrů, čímž je zajištěna větší rozmanitost a schopnost modelu generalizovat. Testovací množina obsahuje jednu skladbu, která slouží pro nezávislé vyhodnocení výkonnosti modelu na datech, která nebyla během učení použita.

Ukázka 9 - kód načtení a sjednocení všech vstupních a výstupních dat (autor, 2025)

```
x_train = []
y_train = []
x_test = []
y_test = []

for track_idx in tracks_train:
    x, y = load_and_process(track_idx)
    x_train.append(x)
    y_train.append(y)

for track_idx in tracks_test:
    x, y = load_and_process(track_idx)
    x_test.append(x)
    y_test.append(y)

x_train = np.vstack(x_train)
x_test = np.vstack(x_test)
y_train = np.vstack(y_train)
y_test = np.vstack(y_test)
x_train = np.expand_dims(x_train, axis=1) # shape (rows, 1,
BINS)
x_test = np.expand_dims(x_test, axis=1)
y_train = np.expand_dims(y_train, axis=1) # shape (rows, 1,
BINS)
y_test = np.expand_dims(y_test, axis=1)
```

Po výběru konkrétních skladeb pro trénink a testování jsou jednotlivé skladby postupně zpracovávány pomocí dříve popsané funkce `load_and_process`. Vstupy i výstupy ze všech skladeb jsou ukládány do seznamů a následně spojeny do jedné velké trénovací a testovací množiny pomocí funkce `np.vstack`, čímž vznikají spojité matice o rozměrech odpovídajících všem sloučeným snímkům. Následně je přidán nový rozměr s velikostí 1 pomocí `np.expand_dims`, aby vstupní i výstupní data odpovídala očekávanému tvaru vstupu konvoluční neuronové sítě, tj. (počet_snímků, 1, 3 * 88).

2.4 Architektura, učení a použití modelu NN

2.4.1 Výběr modelu a motivace architektury

Cílem neuronové sítě je na základě časově-frekvenční reprezentace hudebního signálu určit, které hudební tóny (v podobě MIDI čísel) znějí v jednotlivých časových rámcích. Tento úkol představuje víceznačkovou (multi-label) klasifikaci, protože v jednom časovém okamžiku může znít více tónů současně.

Pro řešení úlohy byla zvolena jednoduchá konvoluční neuronová síť s jednou vrstvou typu 1D Convolution. Tento typ architektury je vhodný pro zpracování sekvenčních dat a dokáže zachytit lokální časové souvislosti mezi sousedními snímky. Díky nízké výpočetní náročnosti je možné síť efektivně trénovat i na běžném hardwaru, což bylo důležité z hlediska dostupných prostředků a rozsahu bakalářské práce.

Vstupní data pro neuronovou síť vznikají spojením tří různých typů spektrálních reprezentací: CQT, STFT a harmonicky rozšířeného CQT. Každá z těchto reprezentací poskytuje odlišný pohled na zvukový signál. CQT nabízí hudebně laděné rozlišení vhodné pro analýzu tónů, STFT lépe zachycuje celkové spektrální charakteristiky a amplitudové změny a HCQT pomáhá zvýraznit harmonicky související frekvence, což může napomoci přesnějšímu rozpoznání základních tónů. Kombinací těchto tří pohledů dochází k lepší generalizaci a vyšší robustnosti modelu vůči různým typům hudebních signálů.

Vzhledem k tomu, že cílem práce není vytvořit konkurenční model vůči nejmodernějším přístupům, ale ukázat princip fungování transkripce hudby pomocí neuronové sítě, byla architektura navržena s důrazem na jednoduchost a srozumitelnost. Takový přístup umožňuje snadnější experimentování, ladění a následnou úpravu architektury bez zbytečné složitosti, což je vhodné pro prototypovou implementaci v rámci bakalářské práce.

2.4.2 Popis architektury NN

Pro úlohu transkripce hudby byla navržena sekvenční architektura neuronové sítě pomocí knihovny TensorFlow zobrazena v ukázce 10. Vstupem modelu jsou časové rámce s 264 vstupními prvky, což odpovídá sloučení tří různých reprezentací (CQT, piano STFT a harmonická CQT), každá s 88 hodnotami odpovídajícími rozsahu klaviatury.

Ukázka 10 - architektura modelu NN pomocí knihovny TensorFlow s Keras API (autor, 2025)

```
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape = (None, 88*3)),
    tf.keras.layers.Conv1D(128, kernel_size=3,
activation='relu', padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(256,
return_sequences=True, dropout=0.3, recurrent_dropout=0.2)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128,
return_sequences=True, dropout=0.3, recurrent_dropout=0.2)),
    tf.keras.layers.Dense(88, activation='sigmoid')
])
```

Vstupní vrstva `tf.keras.layers.Input(shape = (None, 88*3))` definuje tvar vstupních dat. Dimenze `(None, 88*3)` znamená, že model očekává libovolně dlouhou sekvenci časových rámců (osa `None`), přičemž každý rámec obsahuje 264 vstupních hodnot.

Následující vrstva `tf.keras.layers.Conv1D(128, kernel_size=3, activation='relu', padding='same')` provádí jednorozměrnou konvoluci přes časové rámce. Má 128 filtrů (tj. výstupních kanálů) a kernel o velikosti 3, což znamená, že pro každý rámec bere v úvahu i jeho dva sousedy (dohromady 3 rámce). Použití konvoluce umožňuje modelu zachytit lokální časové souvislosti v hudbě. Parametr `activation='relu'` specifikuje použití rektifikované lineární aktivační funkce, která podporuje nelinearitu a omezuje saturaci neuronů. Parametr `padding='same'` znamená, že výstup bude mít stejnou délku jako vstup, protože se vstup doplňuje nulami.

Vrstva `tf.keras.layers.BatchNormalization()` normalizuje aktivace výstupu z předchozí vrstvy (konvoluce) na každém mini-batchi. To pomáhá zrychlit trénování, stabilizovat učení a omezit problémy s explodujícím nebo mizejícím gradientem. Normalizace navíc umožňuje použít vyšší učicí rychlost.

Následují dvě vrstvy typu LSTM (Long Short-Term Memory), které jsou navíc obousměrné (Bidirectional). Každá taková vrstva se skládá ze dvou LSTM podsítí – jedna zpracovává sekvenci dopředu (v čase), druhá zpětně. Díky tomu má model přístup jak k předchozím, tak i k budoucím rámcům v dané sekvenci, což je velmi výhodné zejména u hudby, kde kontext hraje zásadní roli. První LSTM vrstva má 256 jednotek (v každém směru), druhá 128. Parametr `return_sequences=True` znamená, že vrstva vrací výstup pro každý časový krok (ne pouze poslední). Parametr `dropout=0.3` aplikuje dropout na vstupy LSTM – náhodně vynuluje 30 % vstupních jednotek během trénování. Parametr `recurrent_dropout=0.2` náhodně vynuluje 20 % rekurentních spojení (tj. mezičasových závislostí), čímž dále pomáhá regularizaci.

Závěrečná, plně propojená vrstva `tf.keras.layers.Dense(88, activation='sigmoid')` má 88 neuronů, tedy jeden pro každou klávesu. Použitá aktivační funkce sigmoid převádí výstupy na hodnoty mezi 0 a 1, což lze interpretovat jako pravděpodobnost, že daná nota zní ve zpracovaném rámci.

Výsledkem je tedy sekvence vektorů délky 88, kde každá hodnota signalizuje pravděpodobnost aktivace příslušné MIDI noty. Tato forma výstupu odpovídá binární matici o velikosti (počet rámců, 88), která je následně použita pro výpočet ztráty a další fáze trénování.

2.4.3 Proces tréninku a použití modelu NN

Po definování architektury neuronové sítě následuje fáze jejího trénování. V této fázi se model učí na základě trénovacích dat rozpoznávat vztah mezi vstupem (HCQT, CQT a STFT) a odpovídajícím výstupem ve formě binární matice reprezentující výskyt tónů v čase.

Ukázka 11 - kód kompilující model NN (autor, 2025)

```
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

Trénování začíná kompilací modelu, kde se nastavuje optimalizační algoritmus, ztrátová funkce a metrika pro vyhodnocení výkonu během učení, jak je zobrazeno v ukázce 11. V kódu kompilace modelu je několik parametrů:

- `optimizer = 'adam'` – adam (Adaptive Moment Estimation) je populární optimalizační algoritmus, který kombinuje výhody metod AdaGrad a RMSProp. Upravuje rychlost učení jednotlivých vah na základě prvního a druhého momentu gradientu. Je vhodný pro komplexní modely a dobře funguje bez rozsáhlého ladění.
- `loss = 'binary_crossentropy'` – binární křížová entropie je vhodná volba, pokud výstupem modelu je binární matice – tedy pro každý výstupový neuron (odpovídající konkrétnímu tónu) se rozhoduje, zda je aktivní (1), nebo neaktivní (0).
- `metrics = ['accuracy']` – metrika přesnosti zde udává podíl správně klasifikovaných binárních výstupů. I když může být v kontextu nevyvážených dat (většina nul) poněkud zavádějící, slouží jako základní indikátor kvality během trénování.

Po kompilaci modelu následuje samotný proces učení pomocí metody `fit()` jako v ukázce 12, která provádí iterativní úpravu vah modelu na základě vstupních a výstupních trénovacích dat.

Ukázka 12 - kód pro trénink modelu NN (autor, 2025)

```
history = model.fit(x_train, y_train,
          epochs = 5,
          validation_data = (x_test, y_test),
          batch_size = 256)
```

Jako vstupy slouží `x_train` a `y_train`, tedy normalizovaná trénovací data a odpovídající binární výstupní matice reprezentující výskyt tónů v čase. Proces učení je nastaven na pět epoch, což znamená, že model projde celou trénovací množinu pětkrát. Tento počet je zvolen jako kompromis mezi rychlostí trénování a dosažením základní úrovně přesnosti. Kromě toho je definována validační množina (`x_test`, `y_test`), na které je model po každé epoše vyhodnocen, což umožňuje sledování jeho schopnosti zobecňovat a detekovat případné přetrénování.

Velikost batch, tedy počet vzorků zpracovaných v jednom kroku aktualizace vah, je nastavena na 256. To umožňuje efektivní využití paměti a stabilnější konvergenci modelu. Výsledkem volání funkce `fit()` je objekt `history`, který obsahuje záznam průběhu hodnot ztrátové funkce i přesnosti na trénovací a validační množině. Tyto informace lze dále využít k vizualizaci a analýze výkonu modelu během trénování.

Ukázka 13 - kód evaluace modelu a uložení predikce do .csv souboru pro další zpracování (autor, 2025)

```
loss, acc = model.evaluate(x_test, y_test)
print("ACC: ", acc, "LOSS: ", loss)

prediction = model.predict(x_test)
np.savetxt("prediction_float.csv", prediction.squeeze(),
delimiter=",")
prediction = (prediction > 0.5).astype(int)
print("PREC: ", precision_score(y_test.flatten(),
prediction.flatten()),
      "RECALL: ", recall_score(y_test.flatten(),
prediction.flatten()),
      "F1: ", f1_score(y_test.flatten(),
prediction.flatten()))
np.savetxt("y_test.csv", y_test.squeeze(), delimiter=",")
np.savetxt("prediction_bin.csv", prediction.squeeze(),
delimiter=",")
```

Po dokončení trénování modelu je nezbytné ověřit jeho výkon na testovací množině dat pomocí kódu v ukázce 13. K tomu slouží metoda `evaluate()`, která vypočítá ztrátovou funkci a přesnost na základě dříve neviděných testovacích dat. Tyto dvě metriky jsou následně vypsány do konzole, aby poskytly základní přehled o úspěšnosti modelu.

Následuje samotná predikce výstupních hodnot na testovacích datech pomocí metody `predict()`. Výstupem je matice hodnot v rozsahu od 0 do 1, odpovídající pravděpodobnosti výskytu dané noty v konkrétním časovém rámci. Pro další zpracování je tento výstup uložen do souboru `prediction_float.csv`.

Pro vyhodnocení klasifikačních metrik je nutné převést predikované hodnoty na binární podobu. To se provádí pomocí prahování, kde se všechny hodnoty nad 0.5 považují za 1 (nota přítomna), ostatní za 0 (nota chybí). Výsledná binární predikce je uložena do souboru `prediction_bin.csv`.

Pro přesnější posouzení schopnosti modelu detekovat jednotlivé tóny jsou následně vypočteny klasifikační metriky: preciznost (precision), úplnost (recall) a F1 skóre, které je harmonickým průměrem předchozích dvou. Tyto metriky poskytují hlubší vhled do kvality rozpoznávání přítomnosti tónů ve skladbě a jsou rovněž vypsány do konzole. Pro porovnání je do souboru `y_test.csv` uložen i skutečný binární výstup testovacích dat, který je získán z datasetu.

Výstupní soubor `prediction_bin.csv`, který obsahuje binární predikce přítomnosti jednotlivých tónů v čase, slouží jako vstup pro následný skript, který provádí samotný převod do formátu MIDI. Tento soubor tedy představuje klíčový mezikrok mezi výstupem neuronové sítě a finální hudební transkripcí.

2.5 Transkripce predikce do MIDI formátu

2.5.1 Skript pro generování MIDI souboru

Na začátku skriptu v ukázce 14 jsou nastaveny základní parametry pro přepočet časových hodnot, SR a hop length.

Ukázka 14 - kód nastavení parametrů převodu do MIDI a načtení predikce NN (autor, 2025)

```
hop_length = 512
sr = 44100


y_pred = np.loadtxt('prediction_bin.csv', delimiter=',')
```

Následně je pomocí funkce `np.loadtxt()` načten soubor `prediction_bin.csv`, který obsahuje binární predikci neuronové sítě – tedy matice, ve které každý řádek odpovídá jednomu časovému rámci a každý sloupec příslušné tónové výšce. Hodnota 1 na dané pozici značí, že síť predikovala přítomnost konkrétního tónu v daném časovém rámci.

Ukázka 15 - kód vytvoření a upravení objektu třídy MIDIFile (autor, 2025)

```
midi = MIDIFile(1)
midi.addTempo(0, 0, 90)
midi.addProgramChange(0, 0, 0, 74) #piano - 0, flute - 74
velocity = 90
```

Následně je pomocí kódu v ukázce 15 vytvořen nový objekt třídy MIDIFile s jednou stopou `MIDIFile(1)`. Do této stopy je pomocí metody `addTempo()` nastaveno tempo skladby na 90 BPM (úderů za minutu). Pomocí metody `addProgramChange()` se dále definuje zvuk nástroje, který bude použit při přehrávání. Ve skriptu je zvolen nástroj s číslem 74, což odpovídá zvuku flétny podle General MIDI specifikace. Tento nástroj lze snadno změnit např. na klavír (program číslo 0) úpravou posledního parametru. Proměnná `velocity` určuje sílu (hlasitost) každého tónu a je nastavena na hodnotu 90, což reprezentuje středně silnou artikulaci tónu v MIDI standardu.

Ukázka 16 - kód pro vytvoření MIDI souboru z predikce NN (autor, 2025)

```
time_per_frame = hop_length / sr
active_notes = {}
for time_idx in range(y_pred.shape[0]):
    current_time = time_idx * time_per_frame
    for midi_pitch in range(21, 109):
        note_active = y_pred[time_idx, midi_pitch - 21] == 1
        if note_active:
            if midi_pitch not in active_notes:
                active_notes[midi_pitch] = current_time
        else:
            if midi_pitch in active_notes:
                start_time = active_notes.pop(midi_pitch)
                duration = current_time - start_time
                midi.addNote(track=0, channel=0,
pitch=midi_pitch,
                             time=start_time,
duration=duration, volume=velocity)
with open('output.mid', 'wb') as f:
    midi.writeFile(f)
```

V části skriptu zobrazené v ukázce 16 dochází ke zpracování binárních predikcí a jejich převodu do formátu MIDI. Nejdříve se určí délka jednoho rámce v sekundách (`time_per_frame`), což umožňuje převod časových indexů na skutečný čas. Následně se vytvoří slovník `active_notes`, který slouží k dočasnému uchovávání právě znějících tónů spolu s časem jejich začátku.

Pomocí dvou vnořených smyček se iteruje přes časové rámce a všechny možné MIDI noty v rozsahu běžného piana (21–108). Pro každou notu a časový rámec se z binární matice predikcí (`y_pred`) zjistí, zda daná nota právě zní. Pokud nota začne znít a dosud nebyla aktivní, její začátek se zaznamená do slovníku. Naopak pokud přestane znít a byla dříve aktivní, vypočítá se její délka a přidá se do MIDI souboru pomocí metody `addNote`.

Po zpracování všech časových rámců je výsledek zapsán do výstupního souboru `output.mid`, který obsahuje přepis skladby ve formátu MIDI.

2.6 Vyhodnocení výsledků a kvality transkripce

2.6.1 Vyhodnocení metrik během testování

V průběhu vývoje modelu bylo provedeno několik experimentů s cílem zlepšit přesnost predikce a dosáhnout co nejvěrnější transkripce hudebního záznamu. Postupně byly testovány různé kombinace parametrů učení, zejména velikost trénovací dávky (batch size), počet epoch a vážení tříd výstupu. Výsledky ukázaly, že zásadním faktorem ovlivňujícím kvalitu výstupu je především rozsah a rozmanitost trénovacích dat.

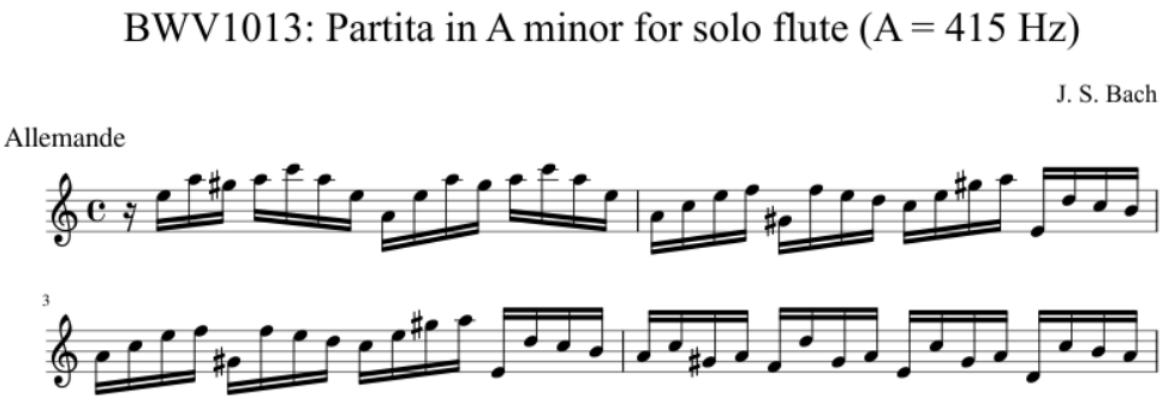
Nejlepších výsledků bylo dosaženo při trénování na 32 skladbách bez váhového vyvažování tříd, s velikostí dávky 512 a trváním 25 epoch. V tomto nastavení model dosáhl následujících metrik:

- Přesnost (accuracy): 76 %,
- Ztrátová funkce (loss): 0,018,
- Precision: 77,3 %,
- Recall: 74 %,
- F1 skóre: 75,6 %.

Experimenty potvrdily, že výkonnost neuronové sítě závisí nejen na její architektuře, ale především na kvalitě trénovacích dat a vhodném nastavení parametrů učení. Ačkoliv byly v průběhu vývoje modelu zkoušeny i kratší tréninky s menším počtem epoch, tyto varianty nevedly ke zlepšení výsledků. Delší trénink (25 epoch) se ukázal jako stabilní a dostatečný pro dosažení požadované kvality výstupu.

2.6.2 Vizualizace, úpravy a vyhodnocení transkripce

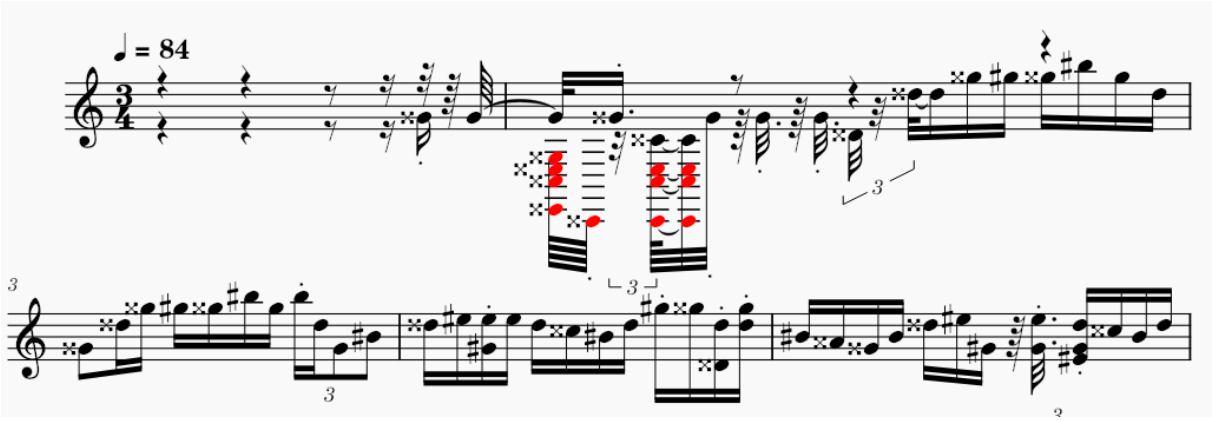
Pro lepší ilustraci výsledků modelu byla vytvořena tato část, ve které je porovnána originální notace skladby, její automatická transkripce generovaná neuronovou sítí a následně upravená verze transkripce, kterou provedl autor na základě hudebního sluchu a odborné znalosti hudební teorie.



Obrázek 20 - ukázka začátku 1. věty Allamande ze skladby Partita in A minor, autor skladby: J. S. Bach, aranž: tetractys (Tetractys, 2024)

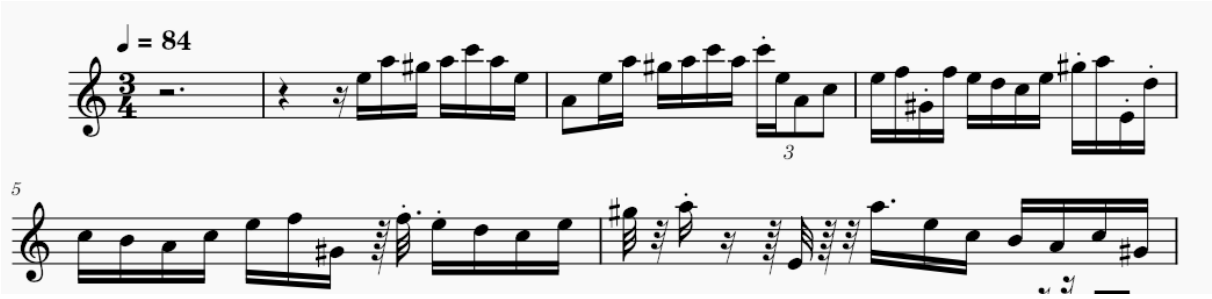
Automaticky vygenerovaná notace modelu sice zachycuje výšky tónů a jejich přibližné časové umístění, nicméně v neupravené podobě často neodpovídá hudebně přirozenému zápisu, který je na obrázku 20. Model totiž pracuje s časovým rozdělením na pevné rámce, což může vést ke vzniku rytmicky nepřesných nebo nehudebně zapsaných pasáží, které působí neohrabaně či nečitelně z pohledu interpreta. Hudební logika, jako je frázování, rytmická pružnost nebo kontextová interpretace not, není neuronovou sítí plně zachycena.

I přes zmíněné chyby a nepřesnosti automatického systému se zvolení časového přístupu zdálo jako lepší řešení. Důvodem je, že skladby na nahrávkách nejsou vždy hrány přesně podle metronomického tempa. V praxi dochází ke zrychlování a zpomalování (tzv. rubato), což by při použití přístupu založeného výhradně na taktech či dobách mohlo vést k nepřesnostem v detekci tónů. Časový přístup se ukázal jako logičtější a univerzálnější řešení pro práci s reálnými nahrávkami.



Obrázek 21 - automatický vygenerovaná transkripce 1. věty Allamande ze skladby Partita in A minor od J. S. Bacha (autor, 2025)

Na obrázku 21 jsou patrné nejčastější chyby, které se v automatickém přepisu po celou dobu testování projevovaly. Na konci prvního a v první polovině druhého taktu se ukazuje jedna z nejčetnějších chyb automatické transkripce, přepis šumu nahrávky do reálně neexistujících not v zápisu. V druhé polovině druhého taktu je vidět další často identifikovaná nepřesnost systému, a to enharmonické záměny, které byly vysvětleny na začátku bakalářské práce. Tyto záměny by se takto hojně v klasických hudebních zápisech nevyskytovaly.



Obrázek 22 - ruční korekce automatické transkripce 1. věty Allamande ze skladby Partita in A minor od J. S. Bacha (autor, 2025)

Po ruční korekci autor dosáhl výsledků, které se již velmi přibližují originální notaci. Úpravy se týkaly především rytmického zarovnání tónů, oprav zjevných nesrovnalostí a zejména enharmonických záměn. Systém totiž nerozlišuje mezi enharmonicky stejnými tóny (např. tón a je totožný s tónem g♯) a někdy zvolí variantu, která z hudebního hlediska nedává smysl vzhledem k tónině nebo okolní harmonii. Tyto případy byly autorem upraveny tak, aby výsledná notace odpovídala běžné hudební praxi.

Tato ukázka slouží nejen k ověření technické funkčnosti modelu, ale také k posouzení jeho praktické použitelnosti z pohledu hudebníka.

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout a realizovat systém pro automatickou transkripci hudebního záznamu pomocí neuronových sítí. Konkrétně se jednalo o detekci výšky tónů a jejich časového umístění v rámci audio nahrávky, přičemž výstupem měla být notová reprezentace ve formátu MIDI. Obsahem bakalářské práce je návrh datového zpracování, implementace modelu strojového učení, jeho vyhodnocení a praktické porovnání výstupu systému s reálnou notací.

V teoretické části byla představena problematika hudebního zápisu, principy frekvenční analýzy a způsoby reprezentace hudby v digitální podobě. Dále byl popsán princip zobrazení CQT a STFT, které se staly hlavním vstupem neuronové sítě. Součástí byla i architektura neuronových sítí vhodných pro zpracování časově-frekvenčních dat.

Praktická část byla zaměřena na návrh samotného modelu a zpracování dat. Jako vstupní reprezentace byla zvolena matice frekvenčně analyzovaných dat, která pokrývala celý standardní klavírní rozsah. Výstupem modelu byla binární matice s rozměrem (počet snímků, 88 tónů), kde každá jednička reprezentovala přítomnost daného tónu v konkrétním čase.

V průběhu vývoje modelu probíhalo několik experimentů, jejichž cílem bylo optimalizovat přesnost predikce. Byly vyzkoušeny různé velikosti dávky (batch size), délky trénování (počet epoch), použití váhování tříd a změny v počtu trénovacích skladeb. Výsledky ukázaly, že jedním z klíčových faktorů byla především velikost a rozmanitost trénovacího datasetu. Zatímco model trénovaný pouze na deseti skladbách dosáhl F1 skóre okolo 0,53, po rozšíření trénovacích dat a vypuštění vyvažování tříd se podařilo dosáhnout F1 skóre až 0,74, což představuje významné zlepšení.

Velmi důležitým aspektem byla také praktická použitelnost výsledků. I když systém dokázal rozpoznávat jednotlivé tóny s poměrně vysokou přesností, samotná výstupní notace nebyla bez dodatečné úpravy hudebně zcela smysluplná. To bylo způsobeno tím, že systém segmentuje signál do pevných časových rámců, což vede k rytmicky nepravidelnému zápisu. Také docházelo k častým enharmonickým záměnám, které jsou pro hudebníka neakceptovatelné, přestože jsou harmonicky a melodicky správné. Ruční úpravy těchto nesrovnalostí vedly k výsledku, který se již přibližoval originálu a byl dobře čitelný i interpretovatelný.

Výsledky bakalářské práce ukazují, že neuronové sítě mají značný potenciál v oblasti automatické hudební transkripce. Přesto je nutné predikovat, že výstup systému bude ve většině případů vyžadovat dodatečnou korekturu ze strany hudebníka nebo editora. Z pohledu automatizace lze však považovat dosažené výsledky za úspěšné!

Přínosem práce je nejen samotný model a jeho implementace, ale i navržený postup zpracování dat, konverze výstupů do MIDI formátu a způsob vyhodnocení výsledků. Tento přístup může být v budoucnu rozšířen například o vícevrstvé zpracování rytmu, začlenění LSTM nebo transformer architektur, případně použití pokročilejších metod postprocessingu pro převod binární predikce do hudebně konzistentního zápisu.

Závěrem lze konstatovat, že cíl práce byl splněn – vytvořený model je schopen převádět hudební nahrávky do notové podoby s vysokou úspěšností. Bakalářská práce může sloužit jako podklad pro další výzkum v oblasti hudební analýzy, automatického zápisu skladeb, nebo dokonce jako podpůrný nástroj pro hudebníky při přepisu vlastních kompozic.

POUŽITÁ LITERATURA

1. AWAN, Abid Ali, 2022. *Recurrent Neural Network Tutorial (RNN)*. Online. DataCamp. Mar 16, 2022. Dostupné z: https://www.datacamp.com/tutorial/tutorial-for-recurrent-neural-network. [cit. 2025-04-29].
2. BADMAN, Annie a KOSINSKI, Matthew, 2025. *What is a dataset?* Online. IBM. Dostupné z: https://www.ibm.com/think/topics/dataset. [cit. 2025-04-29].
3. BROWN, Judith C., 1991. Calculation of a constant Q spectral transform. Online. *The Journal of the Acoustical Society of America*. Roč. 89, č. 1, s. 425-434. Dostupné z: https://www.ee.columbia.edu/~dpwe/papers/Brown91-cqt.pdf. [cit. 2025-04-25].
4. BUHL, Nikolaj, 2023. *F1 Score in Machine Learning*. Online. Encord. July 18, 2023. Dostupné z: https://encord.com/blog/f1-score-in-machine-learning/#:~:text=The%20F1%20score%20or%20F,the%20reliability%20of%20a%20model.. [cit. 2025-05-01].
5. GEEKSFORGEEKS, 20 Aug, 2024. *Fourier Transform*. Online. GeeksforGeeks. 20 Aug, 2024. Dostupné z: https://www.geeksforgeeks.org/fourier-transform/. [cit. 2025-04-22].
6. GEEKSFORGEEKS, 2025. *What is a Neural Network?* Online. GeeksforGeeks. Dostupné z: https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/?ref=rp. [cit. 2025-04-30].
7. GOOGLE DEVELOPERS, 2025. *Classification: Accuracy, recall, precision, and related metrics*. Online. Google Machine Learning Education. 2025-03-03. Dostupné z: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall. [cit. 2025-05-01].
8. GORDON, Whitson, 2012. *What's the Difference Between All These Audio Formats, and Which One Should I Use?* Online. LIFEHACKER. July 18, 2012. Dostupné z: https://lifehacker.com/what-s-the-difference-between-all-these-audio-formats-5927052. [cit. 2025-04-18].
9. GOYAL, Deepam a PABLA, B. S., 2015. Condition based maintenance of machine tools—A review. Online. *CIRP Journal of Manufacturing Science and Technology*. Č. 10, s. 24-35. ISSN 1755-5817. Dostupné z: https://www.sciencedirect.com/science/article/abs/pii/S1755581715000309. [cit. 2025-04-25].
10. HARDESTY, Larry, 2017. *Explained: Neural networks*. Online. MIT News. April 14, 2017. Dostupné z: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414. [cit. 2025-04-26].
11. HASS, Jeffrey, 2025. *Chapter Three: MIDI, 2. MIDI Hardware Specification*. Online. Introduction to Computer Music. Dostupné z: https://cmtext.indiana.edu/MIDI/chapter3_midi_hardware.php. [cit. 2025-04-13].

12. HONS, Miloš, 2019. Hudebně teoretické vzdělání v pojetí Karla Janečka. Online. *HUDEBNÍ VĚDA*. Roč. 2019, č. 3. Dostupné z: https://hudebniveda.cz/2019/03/hons-janecek.pdf. [cit. 2025-04-07].

13. JAISWAL, Sejal, 2025. *Multilayer Perceptrons in Machine Learning: A Comprehensive Guide*. Online. DataCamp. Apr 5, 2025. Dostupné z: https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning. [cit. 2025-04-28].

14. JEŽ, Roman, 2017. *Syntetizéry: mýtus, nebo hračka? Všemocné MIDI*. Online. Frontman. 27.10.2017. Dostupné z: https://www.frontman.cz/syntetizery-mytus-nebo-hracka-vsemocne-midi. [cit. 2025-04-13].

15. KEITA, Zoumana, 2023. *An Introduction to Convolutional Neural Networks (CNNs)*. Online. DataCamp. Nov 14, 2023. Dostupné z: https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns. [cit. 2025-04-28].

16. LENAIL, Alexander, 2019. *NN-SVG*. Online. Alexander LeNail. Dostupné z: https://alexlenail.me/NN-SVG/. [cit. 2025-04-27].

17. LOST AND FOUND, 2023. *Common Problems When Training Deep Learning Models and How to overcome Them*. Online. Medium. Jul 6, 2023. Dostupné z: https://medium.com/@lostandfound2654/common-problems-when-training-deep-learning-models-and-how-to-overcome-them-e37d0ac0a13b. [cit. 2025-05-02].

18. LYONS, Richard G., 2011. *Understanding Digital Signal Processing*. 3rd ed. Pearson Education. ISBN 978-0-13-702741-5.

19. NIELSEN, Michael A., 2015. *Neural Networks and Deep Learning*. Online. Determination Press. Dostupné z: http://neuralnetworksanddeeplearning.com/index.html. [cit. 2025-04-27].

20. NTI AUDIO, 2025. *Fast Fourier Transformation FFT*. Online. NTi Audio. Dostupné z: https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft. [cit. 2025-04-21].

21. PAUL EFFMAN MUSIC, 2025. *Yamaha Professional Eb Baritone Saxophone - YBS-62II - Paul Effman Music*. Online. PAUL EFFMAN MUSIC STORE. Dostupné z: https://www.pemusicstore.com/product/suty-2019/yamaha-professional-eb-baritone-saxophone-ybs-62ii/. [cit. 2025-04-11].

22. PAUL EFFMAN MUSIC, 2025. *Yamaha YTR-8335II Xeno Series Bb Trumpet - Paul Effman Music*. Online. PAUL EFFMAN MUSIC STORE. Dostupné z: https://www.pemusicstore.com/product/suty-2019/yamaha-ytr-8335ii-xeno-series-bb-trumpet/. [cit. 2025-04-11].

23. PROFOUS, Martin, 2011. *Hudební nauka pro základní umělecké školy*. Online. Základní umělecká škola Chrudim. Dostupné z: https://www.zuschrudim.cz/zuschrudim/assets/File/HN345.pdf. [cit. 2025-04-08].

24. SAPPHIRE ODAMAKI, 2025. *Ludwig Van Beethoven - Ode To Joy - Piano Arranged*. Online. MuseScore. Dostupné z: https://musescore.com/user/32650321/scores/5655850. [cit. 2025-04-04].

25. STOLET, Jeffrey, 2009. *Electronic Music Interactive v2*. Online. Dostupné z: https://pages.uoregon.edu/emi/index.php. [cit. 2025-04-16].

26. SVETLIK, Joe, 2025. *MP3, AAC, WAV, FLAC: all the audio file formats explained*. Online. WHAT HIFI? Dostupné z: https://www.whathifi.com/advice/mp3-aac-wav-flac-all-the-audio-file-formats-explained. [cit. 2025-04-18].

27. TETRACTYS, 2024. *Partita in A minor: Allemande (BWV 1013; Solo pour une flûte traversière) by J. S. Bach*. Online. MuseScore. May 14, 2024. Dostupné z: https://musescore.com/user/29345460/scores/5630225. [cit. 2025-05-09].

28. TOBIAS.NIENHAUS, 2025. *You'll Be In My Heart — Phil Collins (Drums)*. Online. MuseScore. Dostupné z: https://musescore.com/user/27077701/scores/20628100. [cit. 2025-04-11].

29. TRIVEDI, Yatri, 2016. *What Are the Differences Between MP3, FLAC, and Other Audio Formats?* Online. How-To Geek. Sep 21, 2016. Dostupné z: https://www.howtogeek.com/40465/htg-explains-what-are-the-differences-between-all-those-audio-formats/. [cit. 2025-04-18].

30. TVRZ, Jakub, 2021. Princip dechových nástrojů z pohledu fyziky. Online. In: *StreTech 2022*. Praha: ČVUT, 12.12.2021. ISBN 978-80-01-07006-2. Dostupné z: https://stretech.fs.cvut.cz/2022/sbornik_2022/pdf/113.pdf. [cit. 2025-04-10].

SEZNAM PŘÍLOH

Příloha A: Archiv zdrojových kódů a PyCharm projekt

Příloha A

Příloha k bakalářské práci

ZDROJOVÉ KÓDY AUTOMATICKÉHO SYSTÉMU

HUDEBNÍ TRANSKRIPCE

Tomáš Netolický

OBSAH ARCHIVU

1. Projektová složka systému pro program PyCharm
2. Zdrojové kódy systému ve formátu pdf