Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Domen Mohorčič

Avtomatska transkripcija klavirske glasbe s konvolucijskimi nevronskimi mrežami

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Igor Kononenko
Somentor: doc. dr. Matevž Pesek

Ljubljana, 2021

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Kandidat: Domen Mohorčič

Naslov: Avtomatska transkripcija klavirske glasbe s konvolucijskimi nevronskimi mrežami

Vrsta naloge: Diplomska naloga na univerzitetnem programu prve stopnje Računalništvo in informatika

Mentor: prof. dr. Igor Kononenko

Somentor: doc. dr. Matevž Pesek

Opis:

Avtomatska transkripcija glasbe je algoritmičnen zapis not iz glasbenega posnetka. Kandidat naj analizira trenutno stanje na področju avtomatske transkripcije glasbe s pomočjo strojnega učenja, kjer naj se osredotoči predvsem na konvolucijske nevronske mreže. Preizkusi in primerja naj več različnih arhitektur in velikosti konvolucijskih nevronskih mrež na zbirki posnetkov klavirske glasbe. Pri tem naj poskuša ugotoviti, katera vrsta normalizacije spektrogramov je bolj ustrezna za tovrstno transkripcijo.

Zahvaljujem se svojima mentorjema za vse nasvete in pomoč pri izdelavi te diplomske naloge. Zahvaljujem se tudi svoji punci Niki in družini, ki so mi stali ob strani, razumeli mojo zaposlenost z diplomsko nalogo in me podpirali pri študiju.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
AMT	automatic music transcription	avtomatska transkripcija glasbe
MIR	music information retrieval	pridobivanje informacij iz glasbe
MPE	multi-pitch estimation	ocenjevanje tonskih višin
NMF	non-negative matrix factorization	nenegativna matrična faktorizacija
HMM	hidden Markov models	prikriti markovski modeli
NN	neural network	nevronska mreža
DNN	deep neural network	globoka nevronska mreža
RNN	recurrent neural network	nevronska mreža s povratno zanko
LSTM	long short-term memory	dolgi kratkoročni spomin
CNN	convolutional neural network	konvolucijska nevronska mreža
DFT	discreet Fourier transform	diskretna Fourierova transformacija
CQT	constant Q transformation	transformacija s konstantnim Q
FFT	fast Fourier transform	hitra Fourierova transformacija
STFT	short-time Fourier transform	kratko-časovna Fourierova transformacija
ReLU	rectified linear unit	popravljena linearna enota

Povzetek

Naslov: Avtomatska transkripcija klavirske glasbe s konvolucijskimi nevronskimi mrežami

Avtor: Domen Mohorčič

V diplomski nalogi obravnavamo problem avtomatske transkripcije glasbe z uporabo globokih nevronskih mrež, natančneje s konvolucijskimi nevronskimi mrežami. Avtomatska transkripcija glasbe je postopek zapisa not iz poslušanja glasbenega posnetka. Preučili smo dosedanje pristope in ugotovili pomanjkanje raziskav o velikosti in o obliki posameznih arhitektur globokih modelov. Raziskali smo uspešnost štirih različnih arhitektur konvolucijskih nevronskih mrež na zbirki klavirskih posnetkov MAPS, ki je pogosta izbira za učenje avtomatske transkripcije glasbe. Preučili smo tudi dva različna pristopa normalizacije spektrogramov: standardizacijo in logaritemsko kompresijo. Izkazalo se je, da na uspešnost transkripcije pozitivno vpliva večje število konvolucijskih plasti v nevronski mreži. Prav tako je bila transkripcija na logaritemsko kompresiranih spektrogramih za 10 % uspešnejša od transkripcije na standardiziranih spektrogramih.

Ključne besede: avtomatska transkripcija glasbe, konvolucijska nevronska mreža, klavirska glasba, transformacija s konstantnim Q, logaritemska kompresija.

Abstract

Title: Automatic music transcription of piano music with convolutional neural networks

Author: Domen Mohorčič

In this thesis we explore the problem of automatic music transcription using deep neural networks, more specific convolutional neural networks. Automatic music transcription is a task of writing the sheet music from musical recordings. We analysed previous studies and found that there was a lack of research about the size and the shape of architecture of deep models. We explored the performance of four different architectures of convolutional neural networks on the piano recordings dataset MAPS, which is a common benchmark for learning automatic music transcription. We also compared two different normalization techniques for spectrograms: standardization and the logarithmic compression. We found out that the performance of transcription is highly correlated with the higher number of convolutional layers. Transcription is also 10% more successful with logarithmic compression instead of standardization.

Keywords: automatic music transcription, convolutional neural network, piano music, constant Q transform, logarithmic compression.

Poglavje 1

Uvod

Transkripcija glasbe je postopek, pri katerem se zapiše notni zapis poslušane glasbe. Kljub temu, da gre za relativno enostaven postopek zapisovanja slišanega, imamo ljudje težave pri zapisovanju poslušanih not, pri čimer so profesionalni glasbeniki med najbolj uspešnimi. Postopek transkripcije običajno vključuje osebo, ki večkrat posluša isti posnetek in pri tem poskuša zapisati, katere note sliši. Pri tem pogosto uporabi svoje teoretično glasbeno znanje o kvantiziranih dolžinah not in o tonaliteti glasbe. Naloga postane zahtevnejša, ko se med glasbo spreminjata hitrost ali tonaliteta, mnogo težja pa postane, ko je v posnetku prisotnih več inštrumentov. Takrat se oseba najprej posveti enemu inštrumentu, nato drugemu in tako naprej. Postopek je zelo zamuden, poleg tega pa popolna transkripcija ni vedno mogoča. Pogosto v določenih delih glasbe nekateri inštrumenti preglasijo druge ali se podvajajo in jih je zato težje ali celo nemogoče slišati. Zaradi vseh nevšečnosti se je pojavila ideja o avtomatski transkripciji glasbe, kjer bi namesto človeka delo opravljal računalnik.

Avtomatska transkripcija glasbe (angl. Automatic Music Transcription – AMT) je postopek, pri katerem računalnik zapiše notni zapis poslušane glasbe. AMT je eden izmed pomembnejših problemov pri pridobivanju informacij iz glasbe (angl. Music Information Retrieval – MIR) [19]. Pri AMT se glasbo pogosto najprej pretvori v spektrogram, ki je vizualna predstavi-

tev amplitud frekvenc v času, na katerem se nato uči izbrani model. Pri enoglasni glasbi je transkripcija v primerjavi z večglasno glasbo enostavna. Lahko se uporabi avtokorelacija signala, iz katere se izlušči frekvenca prisotnega tona, imenovana tudi fundamentalna frekvenca. Pri večglasni glasbi pa je problem mnogo težji. Frekvence tonov, ki so hkrati prisotne v signalu, povzročijo kompleksne medsebojne interakcije in prekrivanje višjih harmonikov fundamentalnih frekvenc. Višje harmonske frekvence so načeloma tišje od fundamentalnih frekvenc, vendar tudi to ne drži za vse inštrumente, zato fundamentalnih frekvenc ne moremo določiti z iskanjem najglasnejših frekvenc. Drugi velik problem je variabilnost vhodnega signala glede na tip inštrumenta. Različni inštrumenti različno zvenijo in imajo drugačno barvo tona (angl. timbre), zato se signali in spektrogrami istih skladb, igranih z različnimi inštrumenti, razlikujejo med sabo. AMT pristopi zaradi tega pogosto uporabljajo kompleksne globoke modele, ki so sposobni razlikovati med frekvencami tonov, harmoniki in potencialno različnimi inštrumenti.

Zaradi vseh težav pri transkripciji glasbe popoln model za večglasno transkripcijo glasbe po vsej verjetnosti ne bo nikoli obstajal, lahko pa naredimo zelo dobre AMT modele za posamezne inštrumente. Pri tem lahko dober AMT model z visoko točnostjo zaznavanja not nadomesti glasbenega eksperta, s čimer se prihrani ogromno časa pri transkripciji. Poleg tega se lahko dober AMT model uporabi za digitalizacijo obstoječe posnete glasbe in kasnejše urejanje le-te ali za zapis not improviziranega glasbenega nastopa solista. Dober AMT model lahko tako v zelo kratkem času transkribira in digitalizira ogromne glasbene zbirke, za kar bi glasbeni eksperti lahko porabili več let. AMT modele se lahko uporabi tudi za pomoč pri transkripciji, saj nam lahko kot osnovo nudi, kateri toni so prisotni ob določenem času v posnetku. Glasbeni ekspert si lahko s tem ogromno pomaga, saj potrebuje razbrati samo, kateri toni so pravilni in kateri lažni ter ali kje kaj manjka, zraven pa uporablja svoje glasbeno znanje. Velikokrat pa so AMT modeli zelo veliki, kompleksni in nepraktični za učenje s časovnega vidika, sploh če nimamo dostopa do boljših računalniških sistemov z veliko zmogljivostjo.

Kljub temu da so kompleksnejši modeli po navadi boljši od enostavnejših, slednji niso veliko slabši od njih. Raziskav o manjših AMT modelih nismo zasledili, zato o velikosti ne vemo prav dosti. Večina raziskav naredi en model, nič pa ne poročajo o rezultatih pri različnih velikostih modelov. Zato smo se odločili narediti primerjavo različno velikih AMT modelov. Najbolj uspešne metode danes vključujejo konvolucijske nevronske mreže, zato smo raziskali, kako velikost konvolucijskih nevronskih mrež vpliva na uspešnost transkripcije.

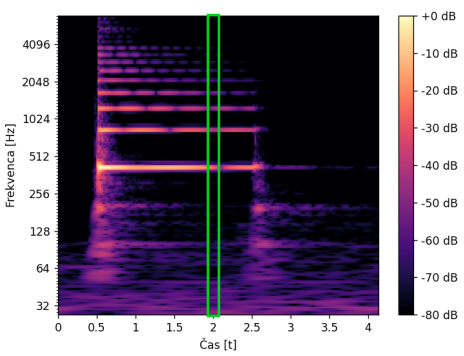
V diplomskem delu se zato posvečamo avtomatski transkripciji večglasne glasbe s konvolucijskimi nevronskimi mrežami in raziščemo odvisnost med velikostjo modelov in njihovimi rezultati. Preverili smo, kako priprava podatkov vpliva na uspešnost učenja modelov in ali večji modeli dosegajo bistveno boljše rezultate. Preverili smo tudi, kako velikost modela vpliva na čas učenja in ali več časa učeni modeli dosegajo boljše rezultate. V poglavju 2 smo naredili pregled dosedanjih poskusov avtomatske transkripcije glasbe in predstavili najbolj uspešne pristope. V poglavju 3 smo predstavili uporabljene metode. V poglavju 4 smo se osredotočili na podatke in implementacijo. V poglavju 5 smo pregledali dobljene rezultate in v poglavju 6 smo rezultate podrobneje analizirali, naredili zaključke in predlagali nadaljnje izboljšave.

Poglavje 2

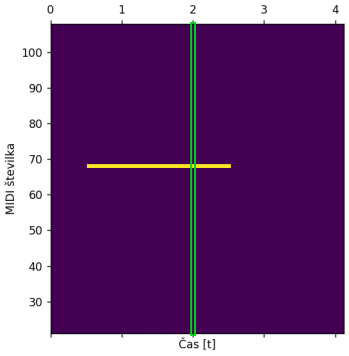
Ozadje

2.1 Avtomatska transkripcija glasbe

Raziskave o avtomatski transkripciji glasbe so se začele v 70-ih letih prejšnjega stoletja in se delijo na štiri kategorije, kot so jih razvrstili Benetos in ostali leta 2019 v [1]: okvirne (angl. frame-level), notne (angl. note-level), inštrumentne (angl. stream-level) in zapisne (angl. notation-level). Okvirna transkripcija (angl. Multi-Pitch Estimation – MPE) se ukvarja z ugotavljanjem prisotnosti posameznih frekvenc v določenem kratkem časovnem intervalu v posnetku, imenovanem okno. Primer okvirne transkripcije je na sliki 2.1. Metode MPE ne poznajo koncepta not in ne morejo ugotoviti kompleksnejših glasbenih lastnosti, kot sta tempo ali tonaliteta. Notna transkripcija (sledenje not, angl. Note Tracking) je nivo višje kot okvirno sledenje. Notna transkripcija nadgradi okvirno transkripcijo tako, da poveže prisotne frekvence skozi čas v note. Note so definirane z začetkom (angl. Onset, start), koncem (angl. Offset, end), višino tona (angl. Pitch), in glasnostjo (angl. Velocity). Naslednja stopnja nad notno transkripcijo je inštrumentna transkripcija. Inštrumentna transkripcija združuje note v skupine, kjer vsaka skupina predstavlja posamezen inštrument. Pri tem si metode pomagajo z dejstvom, da imajo različni inštrumenti različno barvo tona in zato zvenijo različno. Zapisna transkripcija pa poskuša spremeniti posneto glasbo v zapis, ki je lahko berljiv človeku,



(a) CQT spektrogram tona A1 s frekvenco 440 Hz. Na sliki so prisotni tudi višji harmoniki, vendar so časovno in amplitudno šibkejši od fundamentalne frekvence.



(b) Označena prisotnost tonskih višin po času. Ton A1, ki ima MIDI številko 69 in se po MIDI nomenklaturi imenuje tudi A4, je na spektrogramu označen z začetkom okoli 0.5 s in koncem okoli 2.5 s.

Slika 2.1: Pri okvirni transkripciji model kot vhod dobi del spektrograma ali okno, ki je označeno z zelenim okvirjem na levi sliki. Izhod modela je vektor, označen z zelenim okvirjem na desni sliki, ki prikazuje, katere klavirske tipke so bile pritisnjene v oknu. Model z oknom drsi po spektrogramu in za vsako postavitev okna vrne vektor prisotnih tonskih višin.

na primer notni zapis. Taki pristopi potrebujejo kompleksno znanje o glasbenih strukturah, kot so tempo, tonaliteta, ritem in akordi. V tej diplomski nalogi smo se posvetili okvirnim pristopom.

2.2 Predhodniki globokega učenja

Prvi okvirni pristopi so vključevali analizo signalov. Enega prvih pristopov je opisal Moorer leta 1975 v svojem doktorskem delu [21]. V njem je opisal pristop za transkripcijo duetov. Njegov pristop je vključeval frekvenčni detektor in detekor energije signala. Kjer sta se izhoda detektorjev strinjala, da gre za noto, jo je model zapisal. Njegov pristop pa je imel nekaj ključnih

omejitev. V duetu sta morala biti dva različna inštrumenta in igrati sta morala na različnih višinah, da ju je metoda lahko ločila med sabo.

Leta 1989 sta na Osaški univerzi v Tokiju, Japonska, raziskovalca Katayose in Inokuchi kot del raziskave počutja v glasbi naredila model zaznavanja fundamentalnih frekvenc večglasne glasbe [10]. Njun pristop je iz prvih nekaj sekund posnetka razbral dolžino dobe. Nato sta izračunala časovno-frekvenčno sliko in samo ob začetkih dob iskala začetke not. Pri tem je model že imel izračunane vse možne kombinacije večglasij in je tako ob najdenem začetku not samo poiskal pravo kombinacijo. Model je bil pri transkripciji počasen, a vendar uspešen.

Leta 1996 je Hawley v svojem doktorskem delu [7] opisal pristop, ki je vključeval diferencialno analizo spektrogramov. V praksi je to pomenilo računanje odvodov ali razlik med zaporednimi deli spektrogramov. Kjer je bila razlika velika, je model določil začetek note, višino tona pa je dobil iz lokacije, kje na spektrogramu je bila razlika velika. Pristop je bil uspešen, vendar je zaznaval samo začetke not.

Leta 1999 sta Lee in Seung predstavila idejo o nenegativni matrični faktorizaciji (angl. Non-negative Matrix Factorization – NMF) [15], ki sta jo na avtomatsko transkripcijo glasbe prenesla Smaragdis in Brown leta 2003 [27]. V članku sta predstavila uporabo NMF, ki močnostni spekter posnetka glasbe razcepi na dve nenegativni matriki: prva vsebuje frekvence tonov, druga pa časovno komponento tonov. Metoda ni bila popolna, pogosto so manjkali ali pa so bili prisotni dodatni toni.

Leta 2007 so Poliner in ostali v [22] naredili pregled aktualnih pristopov transkripcije glavne melodije, ki je podnaloga avtomatske transkripcije glasbe. Primerjali so pristope sedmih različnih raziskovalcev. Za predprocesiranje so uporabili spektrograme in avtokorelacijo signala. Štirje pristopi so imeli večglasje omejeno, trije pa so imeli potencial zaznati katerokoli večglasje. Večina pristopov je vključevala tudi detektor za začetke not, za čiščenje izhodov pa so uporabili razna pravila in prikrite markovske modele (angl. Hidden Markov Model – HMM). Najboljši pristop, ki je bil uspešen na

različnih tipih glasbe, je pripadal avtorici Dressler iz leta 2005 [5]. Uporabila je spektrograme in analizo signala z modeliranjem sinusoid in dodatnega šuma.

V [29] sta Su in Yang leta 2015 nadgradila pristop analize signala z idejo o detekciji not z najmočnejšo prisotno frekvenco v oknu s sledenjem zaznanih frekvenc skozi naslednja okna. Pristop se je izkazal za uspešnega pri transkripciji večglasne glasbe, še posebej pri prisotnosti pet glasbil ali več.

2.3 Globoko učenje v avtomatski transkripciji glasbe

Na prehodu iz 20. v 21. stoletje so računalniki postali zmogljivejši in mogoča je postala uporaba velikih globokih nevronskih mrež (angl. Deep Neural Network – DNN).

Nevronske mreže (angl. Neural Network – NN) je za namene avtomatske transkripcije glasbe uspešno uporabil Marolt leta 2004 z modelom SONIC [16]. SONIC ima dva dela. Prvi je slušni model, ki s pomočjo adaptivnih oscilatorjev poišče prisotne frekvence. Drugi je zbirka 76 nevronskih mrež, pri čemer je vsaka od njih je zadolžena za detekcijo enega tona. Model ima tudi sistem za detekcijo začetkov not. SONIC je dosegel dobre rezultate, slabo je transkribiral le nizke tihe tone in zelo hitra zaporedja tonov.

Nevronske mreže s povratno zanko (angl. Recurrent Neural Network – RNN) so se v transkripciji pojavile kmalu po tem. Leta 2012 sta Böck in Schedl v [2] za transkripcijo klavirja uporabila plasti za dolgi kratkoročni spomin (angl. Long Short-Term Memory – LSTM), ki so zmožne opisati časovno odvisnost med podatki. Njun model je predstavljal akustični model, ki skrbi za okvirno transkripcijo in hkrati zaznava začetke in višino tonov. Poleg tega je imel model več izhodov, kar je odstranilo potrebo po več modelih za zaznavanje različnih tonov. Izkazal se je za boljšega od tedanjih pristopov, veliko prednost pa je imel pri transkripciji kompleksnejših skladb. Sigtia in ostali so leta 2015 v [26] predstavili model, ki združuje akustični in glasbeno-jezikovni

model, ki skrbi za notno transkripcijo. Slednji je vseboval RNN. Ugotovili so, da predstavljeni hibridni model deluje bolje od ostalih notnih transkripcij. Ugotovili so tudi, da so RNN boljše od NN pri uporabi v akustičnem modelu.

Konvolucijske nevronske mreže (angl. Convolutional Neural Network – CNN) so bile prvič bolj uspešno uporabljene za namene avtomatske transkripcije glasbe leta 2016 v [25], ko so Sigtia, Benetos in Dixon združili pristope Schlüterja in Böcka za detekcijo začetkov not s CNN iz leta 2014 [24] in pristope Humphreya in Belloa za detekcijo akordov s CNN iz leta 2012 [9]. Predlagani model uporablja CNN, ki dosega boljše rezultate od do takrat predlaganih metod v okvirni transkripciji. Celoten postopek s CNN so kasneje istega leta podrobneje prestavili Kelz in ostali v [12], kjer so tudi argumentirali, da je izbira obdelave podatkov ključnega pomena za uspešno transkripcijo glasbe. Primerjali so globoko nevronsko mrežo, konvolucijsko nevronsko mrežo z navadnimi plastmi in konvolucijsko nevronsko mrežo s samimi konvolucijskimi plastmi. Ugotovili so, da konvolucijske plasti pozitivno vplivajo na uspešnost modela, najuspešnejša pa je bila kombinacija konvolucijskih in navadnih plasti.

Hawthorne in ostali so leta 2018 v [8] predlagali nadgraditev uporabe konvolucijskih mrež za okvirno transkripcijo z dodatnim modelom za sledenje not. Njihov model je tako sestavljen iz dveh delov: prvi išče začetke not, drugi pa jim sledi skozi posnetek. Prav tako so CNN nadgradili z RNN, saj lahko zajamejo časovno komponento podatkov, ki jih CNN ne morejo najbolje. Predlagani model je boljši od vseh prejšnjih tako pri okvirni kot notni transkripciji.

Vsi opisani modeli so kompleksni, tako po zgradbi kot po velikosti. Zgrajeni so iz različnih delov z namenom doseganja najboljših rezultatov. Prav tako so večji modeli bolj podvrženi prevelikemu prileganju podatkom (angl. overfitting), kar poslabša njihovo delovanje na nevidenih podatkih in s tem generalizacijo.

Nikjer pa ni nobenega enostavnega pristopa, ki bi raziskal, kako se manjši modeli obnesejo pri transkripciji. Prav tako ni nikjer posebej poudarjen čas

učenja modelov, kar nam da misliti, da ni ravno kratek. Naša motivacija je raziskati, kako uspešni so manjši in enostavnejši modeli in ali se res splača graditi velike in kompleksne modele, če manjši dosegajo primerljive rezultate s krajšim časom učenja.

Poglavje 3

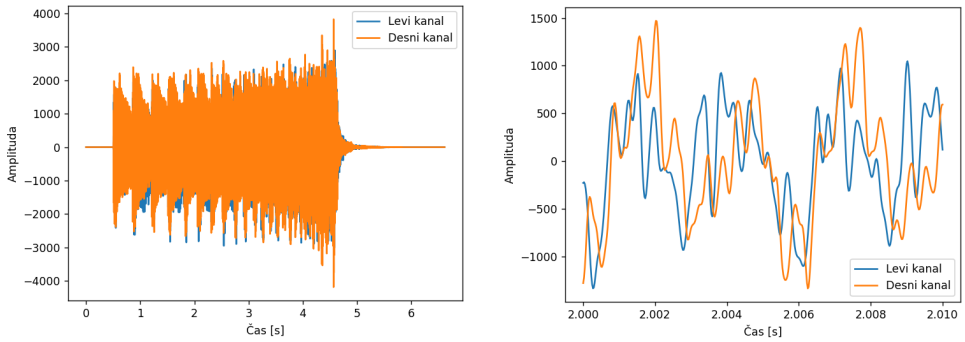
Metode dela

3.1 Oblika podatkov

Pri strojnem učenju potrebujemo velike količine podatkov, ki morajo biti v ustreznem formatu in ustrezno obdelane. V naši diplomski nalogi so podatki o skladbah shranjeni v formatu WAVE, njihov resnični notni zapis pa v formatu MIDI.

Format WAVE (angl. Waveform Audio File Format) je standardni format zapisa glasbe. Datoteka WAVE (s končnico .wav) vsebuje nestisnjen zvočni zapis. Zvok je shranjen na dveh kanalih pri vzorčni frekvenci 44,1 kHz in s 16 biti na vzorec. Zaradi nestisnjenosti in visoke vzorčne frekvence se WAVE format pogosto uporablja za studijske posnetke visoke kakovosti, kar pa je tudi njegova največja slabost, saj so WAVE datoteke pogosto velike in nepraktične za prenos med elektronskimi napravami. Slika 3.1 prikazuje primer vsebine WAVE datoteke iz podatkovne zbirke MAPS, ki je podrobneje predstavljena v razdelku 4.1.

MIDI (angl. Musical Instrument Digital Interface) je standardni komunikacijski protokol za komunikacijo med elektronskimi inštrumenti, računalniki in glasbenimi napravami za namene snemanja, predvajanja in urejanja glasbe. MIDI ima svoj standardni datotečni zapis (s končnico .mid), kamor se lahko zapišejo skladbe. Vsaka skladba ima lahko več inštrumentov, ki so označeni



(a) Slika celotnega WAVE posnetka. Glasba se začne okoli 0.5 s in konča okoli 5 s, posnetek pa je malo daljši od 6 s.

(b) Izrezani 10 ms del iz posnetka levo. Na sliki je prikazano, da je vsebina WAVE datoteke valovna oblika posnetega signala.

Slika 3.1: Prikaz WAVE datoteke posnetka AkPnBcht/ISOL/TR1/MAPS_ISOL_TR1_F_S0_M53_AkPnBcht.wav.

z vrsto. Vsak inštrument vsebuje seznam not, ki jih igra. Note so označene z začetkom, koncem, višino tona in glasnostjo. MIDI datoteke so izjemno uporabne, saj omogočajo enostaven prenos, urejanje in predvajanje glasbenih vsebin. Primer vsebine MIDI datoteke je prikazana na sliki 3.2.

3.2 Spektrogrami

Transkripcija glasbe je naloga, pri kateri se poskuša ugotoviti, kateri toni so prisotni v določenem delu skladbe. Ker so toni primarno definirani z njihovo višino ali frekvenco, si pri transkripciji glasbe lahko pomagamo s spektrogrami. Spektrogram je vizualna predstavitev frekvenčnega prostora signala skozi čas. Frekvence, ki so prisotne v signalu, lahko dobimo tako, da signal pretvorimo iz časovnega v frekvenčni prostor. Vendar pretvorbe ne moremo narediti na celotnem signalu naenkrat, saj tako izgubimo časovno komponento. Rezultat bi bil seznam vseh frekvenc, ki so prisotne v signalu, to pa nam ne koristi pri transkripciji. Pretvorba se mora tako izvesti nad krajšim delom signala, ki mu pravimo okno. Velikost okna določa natančnost fre-

```
Note(start=0.500004, end=0.853560, pitch=53, velocity=92)
Note(start=0.853560, end=1.207116, pitch=54, velocity=92)
Note(start=1.207116, end=1.504412, pitch=53, velocity=92)
Note(start=1.504412, end=1.801708, pitch=54, velocity=92)
Note(start=1.801708, end=2.051703, pitch=53, velocity=92)
Note(start=2.051703, end=2.301699, pitch=54, velocity=92)
Note(start=2.301699, end=2.511917, pitch=53, velocity=92)
Note(start=2.511917, end=2.722135, pitch=54, velocity=92)
Note(start=2.722135, end=2.898907, pitch=53, velocity=92)
Note(start=2.898907, end=3.075678, pitch=54, velocity=92)
Note(start=3.075678, end=3.224326, pitch=53, velocity=92)
Note(start=3.224326, end=3.372974, pitch=54, velocity=92)
Note(start=3.372974, end=3.497978, pitch=53, velocity=92)
Note(start=3.497978, end=3.622983, pitch=54, velocity=92)
Note(start=3.622983, end=3.728099, pitch=53, velocity=92)
Note(start=3.728099, end=3.833214, pitch=54, velocity=92)
Note(start=3.833214, end=3.921607, pitch=53, velocity=92)
Note(start=3.921607, end=4.009999, pitch=54, velocity=92)
Note(start=4.009999, end=4.084323, pitch=53, velocity=92)
Note(start=4.084323, end=4.158647, pitch=54, velocity=92)
Note(start=4.158647, end=4.221143, pitch=53, velocity=92)
Note(start=4.221143, end=4.283638, pitch=54, velocity=92)
Note(start=4.283638, end=4.336196, pitch=53, velocity=92)
Note(start=4.336196, end=4.388754, pitch=54, velocity=92)
Note(start=4.388754, end=4.432943, pitch=53, velocity=92)
Note(start=4.432943, end=4.477133, pitch=54, velocity=92)
Note(start=4.477133, end=4.514295, pitch=53, velocity=92)
Note(start=4.514295, end=4.551457, pitch=54, velocity=92)
Note(start=4.551457, end=4.582705, pitch=53, velocity=92)
Note(start=4.582705, end=4.613952, pitch=54, velocity=92)
```

Slika 3.2: Vsebina MIDI datoteke AkPnBcht/ISOL/TR1/MAPS_ISOL_TR1_F_S0_M53_AkPnBcht.mid. Datoteka vsebuje note, vsaka nota ima začetek, konec, višino in glasnost. Pitch označuje višino tona v MIDI številkah. MIDI številki 53 in 54 označujeta tona mali F in mali FIS.

kvenčnega prostora. Preveliko okno pomeni izgube v časovni komponenti, s premajhnim pa ne zaznamo nizkih frekvenc zaradi premajhnega števila vzorcev, zato moramo velikost okna izbrati previdno. Če okna nato zložimo enega zraven drugega, dobimo sliko frekvenčnega prostora, ki je odvisen od časa. V nadaljevanju si oglejmo dve najpogostejši pretvorbi iz časovnega v frekvenčni prostor: diskretno Fourierovo transformacijo (angl. Discreet Fourier Transform – DFT) in transformacijo s konstantnim Q (angl. Constant Q Transform – CQT).

Diskretna Fourierova transformacija je diskretna oblika Fourierove transformacije, ki lahko transformira poljubno neperiodično funkcijo v kombinacijo sinusoid z različnimi frekvencami, fazami in amplitudami. Naš signal je neperiodična funkcija, zato namesto Fourierove vrste uporabimo Fourierovo transformacijo. Poleg tega signala nimamo podanega kot funkcijo, ampak kot zajeti kvantizirani vzorec, zato uporabimo DFT. Pri tem si lahko pomagamo z algoritmom hitra Fourierova transformacija (angl. Fast Fourier Transform – FFT), ki predstavlja učinkovito implementacijo DFT s časovno kompleksnostjo $O(n \log n)$, kjer n predstavlja število vzorcev v signalu.

Enačba 3.1 predstavlja enačbo DFT. Pri tem v enačbi N predstavlja število zajetih vzorcev ali velikost okna in x_n predstavlja zajeti signal ob časih $k = 0, 1, \ldots, N-1$. X_k je kompleksno število, ki predstavlja amplitudo in fazo sinusoid s frekvenco k/N na časovno enoto, ki je enaka velikosti okna. Zaloga vrednosti enačbe DFT so tako kompleksna števila. Amplitudo signala dobimo z izračunom velikosti kompleksnega števila po formuli $|z| = \sqrt{Im(z)^2 + Re(z)^2}$.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i\,2\pi\,n\,k/N} \tag{3.1}$$

Pri tem smo omejeni z zaznavo frekvenc. Po Nyquist-Snannonovem teoremu ne moremo zaznati frekvenc, ki so večje od polovične frekvence vzorčenja. Ker imamo WAVE datoteke, ki so vzorčene pri 44,1 kHz, DFT ne mora zaznati frekvenc, ki so višje od 22,05 kHz. Najvišji ton na klavirju je C5 (MIDI oznaka C8 ali številka 108) s frekvenco 4186 Hz, kar pomeni, da lahko vse

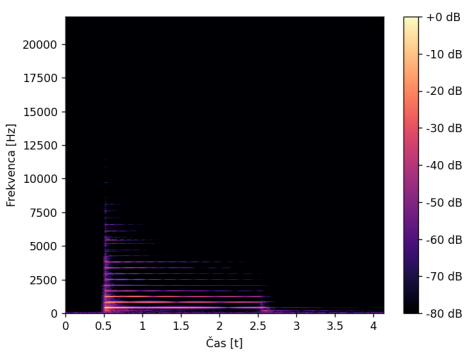
tone klavirja zaznamo.

V praksi se uporablja tudi algoritem kratko-časovna Fourierova transformacija (angl. Short-Time Fourier Transform – STFT), ki je različica FFT algoritma. STFT je narejen za namene raziskovanja, kako se frekvence spreminjajo skozi čas. Algoritem signal sam razdeli na bloke (okna), ki se lahko prekrivajo med sabo, in nad njimi izvede transformacijo. Slika primera izhoda STFT algoritma je predstavljena na sliki 3.3(a). Zaradi preglednosti je na sliki 3.3(b) prikazana povečava istega primera spektrograma.

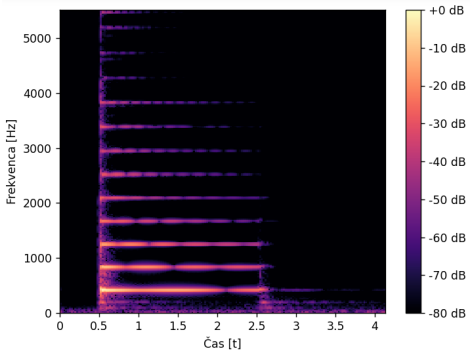
Transformacija s konstantnim Q [3] je podobna DFT z nekaj pomembnimi razlikami. Algoritem CQT uporabi različno velika okna za zaznavo različno visokih frekvenc, medtem ko ima DFT okna konstantne velikosti. Za nizke frekvence uporabi daljše okno, kar nam poveča frekvenčno resolucijo, za visoke frekvence pa uporabi krajše okno, kar nam da visoko časovno resolucijo. S tem algoritem CQT predstavlja dober kompromis med časovno in frekvenčno resolucijo. Druga razlika je uporaba logaritemske lestvice. DFT razdeli frekvenčni prostor linearno, medtem ko CQT uporabi logaritemsko lestvico. Uporaba logaritemske lestvice izhaja iz biološkega vidika zaznavanja zvoka, ki ga zaznavamo logaritemsko. Če ima osnovni ton frekvenco f_0, ima ton, ki je oktavo višje od osnovnega, frekvenco $f_1 = 2f_0$, in ton, ki je dve oktavi nad osnovnim, frekvenco $f_2 = 2f_1 = 4f_0$. Zaradi tega so višji toni v frekvenčnem prostoru vedno bolj narazen, medtem ko so nižji toni vedno bolj skupaj. Ko pa jih poslušamo, imamo občutek, da so narazen linearno. DFT z linearno skalo zato ne razločuje najbolje med nizkimi toni, hkrati pa pušča ogromno prostora med toni z visokimi frekvencami. CQT izkoristi znanje o relaciji med frekvencami tonov in jih zato izračuna na logaritemski lestvici. Slika primera izhoda CQT algoritma je na sliki 3.3(c).

3.3 Modeli učenja

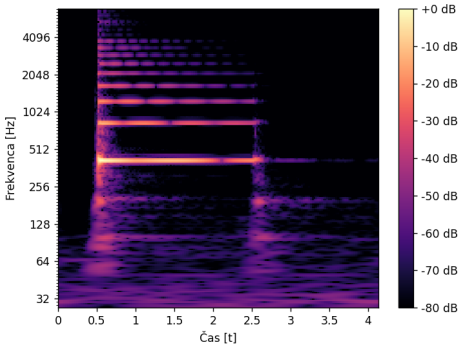
Za avtomatsko transkripcijo glasbe smo implementirali modele z nevronskimi mrežami in konvolucijskimi nevronskimi mrežami.



(a) STFT spektrogram tona A1.

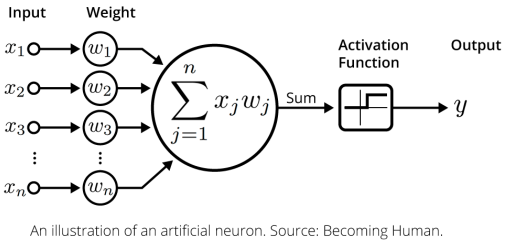


(b) Povečan STFT spektrogram tona A1.



(c) CQT spektrogram tona A1.

Slika 3.3: Slike spektrogramov tona A1 s frekvenco 440 Hz. Na sliki levo je primer STFT spektrograma in na sliki desno je povečava slike levo. Na sliki spodaj je primer CQT spektrograma. CQT spektrogram ima logaritemsko skalo, medtem ko je skala STFT spektrograma linearna. Zato se na CQT spektrogramu lažje vidi prisotne frekvence, medtem ko so na STFT spektrogramu (levo) zelo nepregledne. Sliki levo in spodaj sta neposredna izhoda algoritmov STFT in CQT in nista spremenjeni, frekvenčni osi pa sta samo označeni s pravilnimi vrednostmi.



Slika 3.4: Grafični prikaz enačbe nevrona. Vir: https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english/

Nevronska mreža je model strojnega učenja, ki uporablja več plasti celic ali nevronov, ki so povezani med sabo. Vsaka NN ima vsaj dve plasti, vhodno in izhodno plast, lahko pa ima tudi vmesne plasti, ki se imenujejo skrite. Vsaka plast je sestavljena iz več nevronov, vsi nevroni ene plasti imajo za vhod izhode nevronov prejšnje plasti. Tako dobimo mrežo povezav, ki je zmožna aproksimirati zelo kompleksne relacije/funkcije.

Vsak nevron je definiran kot utežena vsota izhodov prejšnje plasti, nad katerim je izvedena aktivacijska funkcija, kot je to prikazano v enačbi 3.2.

$$x_j^l = f\left(\sum_{k=0}^{n_{l-1}} w_{jk}^l x_k^{l-1} + b_j^l\right) \tag{3.2}$$

Tu x_j^l predstavlja vrednost j-tega nevrona v l-ti plasti, $\sum_{k=0}^{n_{l-1}} w_{jk}^l x_k^{l-1}$ predstavlja uteženo vsoto vrednosti nevronov prejšnje plasti, b_j^l predstavlja pristranskost (angl. bias) nevrona j in f predstavlja aktivacijsko funkcijo. Enačba je grafično prikazana na sliki 3.4.

Najpogosteje uporabljene aktivacijske funkcije so ReLU (angl. Rectified Linear Unit), sigmoidna funkcija (angl. sigmoid) in hiperbolični tangens (tanh). ReLU funkcija omeji izhod nevrona na interval $[0, \infty)$. Pri tem pozitivne vrednosti pusti pri miru in negativne nastavi na 0. Zaradi izpuščanja negativnih vrednosti je računsko enostavna, prav tako omogoča hitrejšo konvergenco do lokalnega minimuma funkcije izgube (angl. loss function). Sigmoidna funkcija, imenovana tudi logistična funkcija, izhod omeji na interval

$(0, 1)$. Funkcija je upobna, ko hočemo izhod interpretirati kot verjetnost, in je odvedljiva, kar poenostavi učenje. Hiperbolični tangens je zelo podoben sigmoidni funkciji, le da izhod omeji na interval $(-1, 1)$. Izhode si tako lahko razlagamo kot negativne, nevtralne ali pozitivne. Ker ima hiperbolični tangens bolj strm odvod kot sigmoidna funkcija, je bolj primeren za hitrejše učenje.

Ker enačba 3.2 velja za vse nevrone v NN, jo lahko zapišemo v matrični obliki v enačbi 3.3.

$$X_l = f\left(W_l X_{l-1} + B_l\right) \tag{3.3}$$

V njej je X_l stolpični vektor, ki predstavlja izhod plasti l. W_l je matrika uteži plasti l in je velikosti $n_{l-1} \times n_l$, kjer n_l predstavlja velikost l-te plasti. B_l predstavlja stolpični vektor pristranskosti plasti l. Med matriko W_l in vektorjem X_{l-1} poteka skalarno množenje.

NN je mogoče učiti zaradi algoritma, ki se imenuje vzvratno razširjanje napake (angl. backpropagation of error) in uporablja gradientni spust. Pri učenju NN se najprej pri določenem vhodu izračunajo vrednosti vseh nevronov in končno tudi vrednosti izhodov, ki se nato primerjajo s pravilnimi izhodi. Iz razlike se nato s pomočjo odvodov izračuna, za koliko je potrebno katere parametre NN spremeniti. Parametri NN so uteži w^l_{jk} in pristranskosti b^l_j vseh nevronov.

Konvolucijska nevronska mreža je mreža, ki je podobna NN z razliko, da implementira konvolucijo. Konvolucija (angl. convolution) je matematična operacija med dvema funkcijama, katere izhod je funkcija, ki opisuje, kako je oblika ene funkcije spremenjena z drugo funkcijo. Pri tem pogosto rečemo, da je druga funkcija filter, ki se izvede na prvi funkciji.

Enačba 3.4 je enačba konvolucije. V enačbi je funkcija f spremenjena s funkcijo g. Izhod je funkcija $(f * g)$. Konvolucija se pogosto uporablja pri obdelavi slik, računalniškem vidu in obdelavi signalov.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau \tag{3.4}$$

CNN so NN s konvolucijskimi plastmi. Pri tem so povezave med plastmi lokalne in deljene med nevroni. Tako je vsak nevron naslednje plasti povezan le z nekaj nevroni prejšnje plasti.

Pri CNN so v uporabi tudi podvzorčne (angl. subsampling, downsampling, pooling) plasti. Te zmanjšajo izhod ene konvolucijske plasti pred vhodom v naslednjo plast. S tem se podatki agregirajo in naslednje plasti dobijo manj podatkov v obdelavo, kar zmanjšuje število parametrov in čas učenja. Podvzorčenje poteka tako, da se iz določene regije izhoda konvolucijske plasti ohrani ali izračuna le ena številka. Pogosti podvzorčni plasti sta Max pooling, ki ohrani samo največjo vrednost določene regije, in Average pooling, ki vrne povprečno vrednost določene regije.

3.4 Metrike

Pri strojnem učenju moramo vedeti, kako dober je model po učenju. Pri tem si pomagamo z metrikami. Metrike so kvantitativne meritve, s katerimi lahko merimo uspešnost, učinkovitost ali lastnost nečesa. V našem primeru metrike na naučenih modelih predstavljajo njihovo lastnost uspešnosti pri napovedovanju na še ne videnih podatkih. Pri tem se najprej določijo osnovne metrike, iz katerih se lahko izračunajo bolj kompleksne: resnično pozitivni (angl. true positive, TP), resnično negativni (angl. true negative, TN), lažno pozitivni (angl. false positive, FP) in lažno negativni (angl. false negative, FN) primeri. TP nam predstavlja število primerov, ki so definirani kot pozitivni in jih je model označil za pozitivne. TN je število negativnih primerov, ki jih je model označil za negativne. FP in FN so primeri, ki jih je model označil ravno nasprotno, kot pa so v resnici. Pri strojnem učenju se najpogosteje uporabljajo metrike točnosti (angl. accuracy), senzitivnosti (angl. sensitivity) in specifičnosti (angl. specificity).

Točnost se pogosto uporablja v binarnih klasifikacijskih primerih, kjer imamo pozitivne in negativne primere. Točnost se izračuna po enačbi 3.5, kjer se število resnično pozitivnih in resnično negativnih primerov deli s

številom vseh primerov. S tem dobimo odstotek, ki nam pove, koliko primerov je model pravilno klasificiral oziroma uvrstil.

$$\text{točnost} = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.5}$$

Senzitivnost nam pove, kolikšen odstotek primerov, ki jih je model označil za pozitivne, je res pozitivnih. Izračuna se po enačbi 3.6, kjer se število resnično pozitivnih primerov deli z vsoto števila resnično pozitivnih in števila lažno negativnih primerov.

$$\text{senzitivnost} = \frac{TP}{TP + FN} \tag{3.6}$$

Specifičnost je mera, ki nam pove, kolikšen odstotek primerov, ki jih je model označil za negativne, je res negativnih. Izračuna se po enačbi 3.7, kjer se število resnično negativnih primerov deli z vsoto števila resnično negativnih in števila lažno pozitivnih primerov.

$$\text{specifičnost} = \frac{TN}{TN + FP} \tag{3.7}$$

Če gledamo na problem transkripcije glasbe kot na klasifikacijski problem, lahko vidimo, da je od zgoraj naštetih mer za nas zanimiva samo senzitivnost. Ker je v skladbah v veliki večini naenkrat prisotnih le nekaj tonov (pozitivni primeri), je tako velika večina primerov negativnih. Model, ki ne bi napovedoval nič, bi tako dosegel visoko točnost in specifičnost, a nizko senzitivnost. Za podatke, kjer je pozitivnih primerov malo, in za potrebe diplomske naloge potrebujemo še metrike preciznosti (angl. percision), priklica (angl. recall) in F vrednosti (angl. F-score).

Preciznost se izračuna po enačbi 3.8, kjer se število resnično pozitivnih primerov deli z vsoto števila resnično pozitivnih in števila lažno pozitivnih primerov.

$$\text{preciznost} = \frac{TP}{TP + FP} \tag{3.8}$$

Priklic se izračuna po enačbi 3.9, kjer se število resnično pozitivnih primerov deli z vsoto števila resnično pozitivnih in števila lažno negativnih primerov. Pomembno je poudariti, da imata priklic in senzitivnost enak pomen in enaki formuli, razlika je le v tem, da ime senzitivnost izvira iz področja statistike, medtem ko ime priklic izvira iz področja informatike.

$$\text{priklic} = \frac{TP}{TP + FN} \tag{3.9}$$

Preciznost nam tako pove, koliko izbranih elementov je ustreznih, priklic pa, koliko ustreznih elementov je bilo izbranih. Pri tem so izbrani elementi tisti, ki jih je model označil za pozitivne, ustrezni pa tisti, ki so zares pozitivni.

F vrednost je harmonična sredina med preciznostjo in priklicem, izračuna se po enačbi 3.10. Opisuje uspešnost binarne klasifikacije, kjer imamo veliko število negativnih primerov. V diplomski nalogi smo jo uporabili za splošno uspešnost modela, kjer višja F vrednost pomeni boljši model.

$$\text{F vrednost} = 2 * \frac{\text{preciznost} * \text{priklic}}{\text{preciznost} + \text{priklic}} \tag{3.10}$$

Poglavje 4

Poskusi

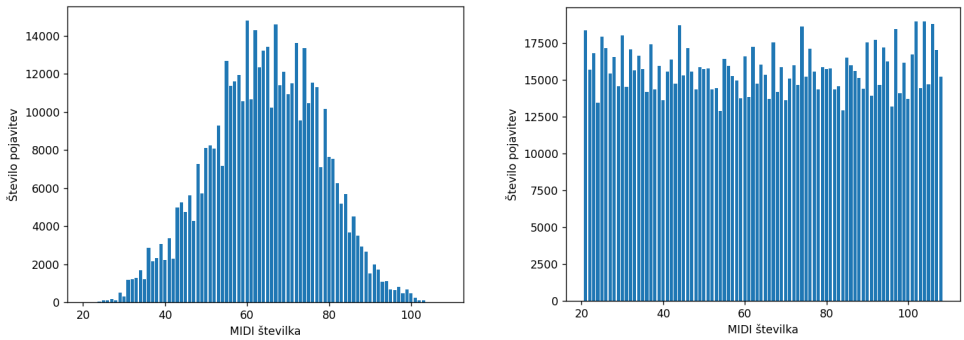
4.1 Podatki in obdelava

Pri diplomski nalogi smo uporabili podatkovno zbirko anotiranih klavirskih skladb MAPS (MIDI Aligned Piano Sounds) [6]. V zbirki so posneti trije virtualni klavirji (The Grand 2 iz programa Steinberg, Akoustik Piano iz programa Native Instruments in The Black Grand iz programa Sampletekk) in en resnični klavir, disklavir. Snemani so bili na različnih krajih, kot so cerkev, koncertna dvorana in studio, skupno je tako v zbirki devet različnih kombinacij klavirjev in snemalnih krajev. Posnetki so shranjeni v WAVE formatu, zraven vsakega je tudi pripadajoča MIDI datoteka. Posnetki posameznih kombinacij klavirja in snemalnega kraja so razdeljeni še na štiri podmnožice. Prva je ISOL, ki vsebuje posnetke posameznih dolgih in kratkih tonov ter kromatičnih lestvic. Druga je RAND, ki vsebuje posnetke akordov z naključnimi toni in različnim številom prisotnih tonov. Tretja je UCHO, ki vsebuje akorde, ki so pogosto prisotni v glasbi zahodnega sveta. Zadnja je MUS, ki vsebuje posnetke klasičnih glasbenih del avtorjev, kot so Bach, Chopin, Grieg, Mozzart, Schubert in drugi. Za učenje modelov smo se odločili, da uporabimo samo podmnožico MUS, saj nas zanima transkripcija pravih skladb in ne posameznih not. Za razdelitev skladb na učne in testne so Sigtia, Benetos in Dixon v [25] predlagali, da se model uči na sintetiziranih sklad-

bah virtualnih klavirjev in testira na resničnih posnetkih. Testne skladbe tako pripadajo posnetkom dveh klavirjev, ki sta označena z ENSTDkAm in ENSTDkCl, vsi ostali posnetki so učni. Tako razdelitev argumentirajo kot smiselno in logično, saj so anotirane skladbe po navadi posnete v nadzorovanem okolju, medtem ko je glasba, ki jo poskušamo transkribirati, posneta v nenadzorovanem in neidealnem okolju.

Pri pregledu zastopanosti not med učnimi MUS skladbami v MAPS zbirki smo opazili neenakomerno porazdelitev prisotnih not, kar je v klasični glasbi običajno, saj ni pogosto slišati zelo visokih ali nizkih tonov. Porazdelitev je normalna, kjer velika večina not pripada srednji legi. Povprečen ton je E1 (MIDI ime E4 in številka 64) s frekvenco 329.63 Hz, standardni odklon pa predstavlja 13.50 tonov, kar je ton in pol več kot ena oktava. S temi podatki lahko vidimo, da približno 95% vseh not med MUS skladbami v MAPS zbirki predstavlja 44 različnih tonov, kar predstavlja polovico vseh tonov, ki jih je možno zaigrati na klavirju. Za boljše učenje smo se odločili, da skladbe transponiramo in tako povečamo zastopanosti vseh tonov. Transpozicija je glasbeni postopek, pri katerem določeno skupino not prestavimo za določeno število tonov višje ali nižje. Pri tem se ohranita melodija in ritem, spremeni pa se višina tonov, kar povzroči, da slišimo isto melodijo, ki je nižja ali višja. Vse učne skladbe v MUS smo transponirali za 30 tonov višje in nižje in tako potrojili število učnih skladb. Testnih skladb se nismo dotikali, saj so bile posnete na resničnih klavirjih. Pri tem smo za generiranje posnetkov uporabili Python knjižnjico midi2audio in njen modul FluidSynth. Po transponiranju smo ponovno pregledali porazdelitev not med učnimi podatki, ki je zdaj veliko bolj uniformna. To daje našemu modelu boljše možnosti, da se nauči vseh tonov in ne samo najbolj pogostih. Distribucija not pred in po transpoziciji se vidi na Sliki 4.1.

Pri pretvorbi skladb iz WAVE datotek v spektrograme smo uporabili CQT namesto STFT, kot to argumentirajo Sigtia, Benetos in Dixon v [25]. Za računanje CQT spektrogramov smo si pomagali s Python knjižnico librosa [18]. Skladbe smo najprej s povprečenjem kanalov pretvorili v enokanalne.



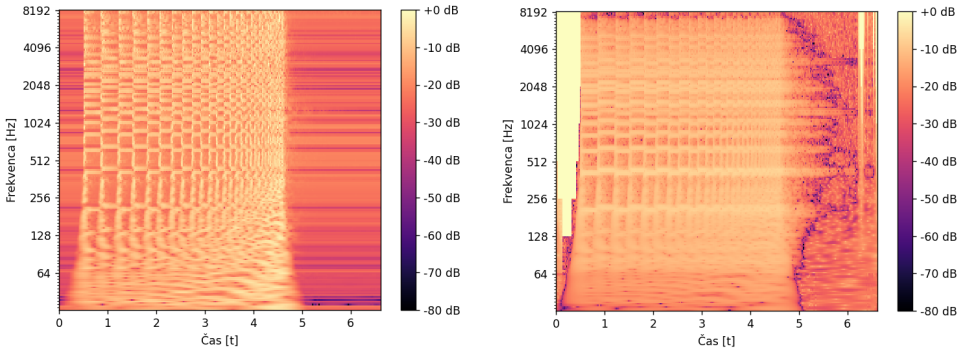
(a) Distribucija not med MUS testnimi skladbami.

(b) Distribucija not po transponiranju MUS skladb.

Slika 4.1: Distribucija not pred in po transpoziciji. Slika levo prikazuje distribucijo not med učnimi skladbami v MUS delu MAPS zbirke. Slika desno pikazuje distribucijo not po transpoziciji vseh MUS učnih skladb za 30 tonov navzgor in navzdol. Pred transpozicijo so note porazdeljene normalno, po transpoziciji pa imajo uniformno porazdelitev.

CQT se je nato izračunal nad 8 oktavami s 24 vrednostmi na oktavo in z velikostjo okna 1024 pri vzorčni frekvenci 44,1 kHz, kar je pomenilo, da je bilo vsako okno dolgo 43,07 ms. Najnižja frekvenca CQT algoritma je bila nastavljena na 27,5 Hz, kar ustreza tonu A subkontra (MIDI ime A0 in številka 21), ki je najnižji ton na klasičnem klavirju. Dobljeni spektrogram je tako imel 192 vrstic, ki so predstavljale frekvence tonov.

Normalizacija spektrogramov je potekala na dva načina. Prvi način je standardizacija. Izračunali smo 192 povprečnih vrednosti in standardnih odklonov za vsako vrstico posebej nad vsemi spektrogrami, ki so pripadali učni množici. Nato smo vsem spektrogramom odšteli vektor povprečnih vrednosti in jih delili s standardnimi odkloni. Standardizacija nam da spektrograme, na katerih se posamezni toni veliko bolje vidijo, vendar pa je ta postopek zelo odvisen od podatkov, ki jih imamo na voljo. Primer standardiziranega spektrograma je na sliki 4.2(a). Drugi način je logaritemska kompresija, ki jo predlagajo Choi in ostali v [4] ter Bőck in Schedl v [2] in je neodvisna od podatkov. Pri logaritemski kompresiji smo izračunali logaritem spektrograma,



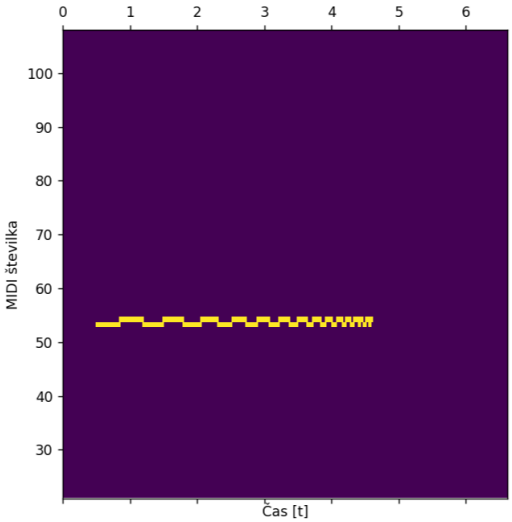
(a) Standardiziran CQT spektrogram.

(b) Logaritemsko kompresiran CQT spektrogram.

Slika 4.2: Normalizirana CQT spektrograma. Na sliki levo je standardiziran spektrogram, na sliki desno pa logaritemsko kompresiran CQT spektrogram. Spektrograma na sliki sta nastala iz posnetka AkPnBcht/ISOL/TR1/MAPS_ISOL_TR1_F_S0_M53_AkPnBcht.wav.

ki smo mu prišteli majhno konstanto zaradi preprečevanja računskih napak ($kompresija = log(spektrogram + \epsilon)$). Choi in ostali argumentirajo uporabo logaritemske kompresije zaradi človeške zaznave glasnosti, ki je logaritemska in ne linearna [20]. Prav tako so pokazali, da so modeli, učeni na logaritemsko kompresiranih podatkih, boljši. Primer logaritemsko kompresiranega spektrograma je na Sliki 4.2(b).

MIDI datoteke so služile kot ustrezen izhod za skladbe, vendar jih je bilo potrebno spraviti v ustrezno obliko. Podobno kot WAVE datoteke so bile pretvorjene v spektrograme, vendar na drugačen način. S pomočjo Python knjižnjice pretty_midi [23] smo prebrali vse note v MIDI datotekah in njihovo prisotnost označili v binarnih matrikah, ki so prevzele vlogo izhodnih spektrogramov. Matrike so imele 88 vrstic in toliko stolpcev, kot jih je imel spektrogram pripadajočega WAVE posnetka. Pri tem smo upoštevali resolucijo spektrogramov, ki je bila 43,07 ms, in z isto frekvenco vzorčili prisotnosti tonov v MIDI datotekah. Primer MIDI spektrograma je na Sliki 4.3.



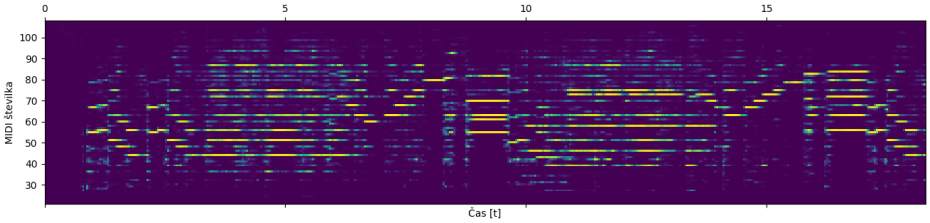
Slika 4.3: Spektrogram, narejen iz MIDI datoteke AkPnBcht/ISOL/TR1/MAPS_ISOL_TR1_F_S0_M53_AkPnBcht.mid. Na njem je prikazano, kaj je zaigrano na pripadajočem WAVE posnetku.

4.2 Arhitekture nevronskih mrež

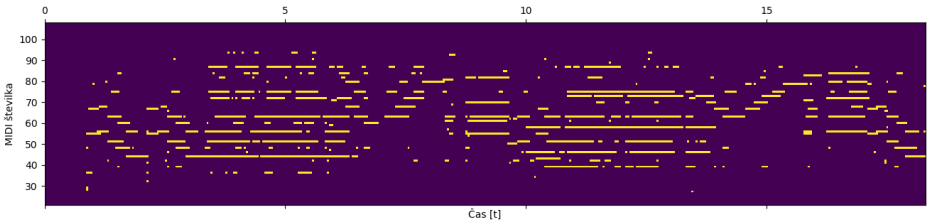
Pri diplomski nalogi smo se osredotočili na raziskovanje, kako velikosti navadnih in konvolucijskih modelov vplivajo na rezultate. Pri grajenju modelov smo si pomagali s Pyhton knjižnicama TensorFlow [17] in Keras.

Vhod modela je bil del spektrograma, visok 192 in širok 5 celic, kar predstavlja 215,35 ms časovni interval. Višina je enaka izhodu CQT algoritma, širina 5 pa je okno, ki predstavlja kontekst določenega dela spektrograma v času. Izhod modela je vektor 88 vrednosti, ki so zaradi logistične aktivacijske funkcije omejene na interval $(0, 1)$, kjer vrednost 1 pomeni pritisnjeno klavirsko tipko in vrednost 0 nepritisnjeno tipko. Interval je odprt in zato vrednosti 0 in 1 nista možni, lahko pa si ga razlagamo kot verjetnost, da je določena tipka pritisnjena. Model napoveduje pritisnjene tipke srednjega (tretjega) stolpca izseka spektrograma. Primer izhoda modela skupaj z resničnim izhodom je prikazan na sliki 4.4.

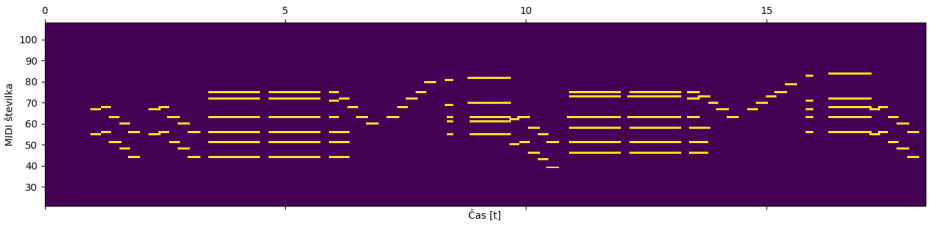
Pri grajenju modelov smo uporabili Dropout [28], ki je tehnika regulari-



(a) Primer izhoda modela. Izhodi so vrednosti na intervalu $(0, 1)$ in si jih razlagamo kot verjetnost, da je določena tipka na klavirju ob določenem času pritisnjena.



(b) Izhod modela, binariziran s pragom 0,5. Rezultat je izhod, ki ga je mogoče primerjati z resničnim spektrogramom, narejenim iz MIDI datoteke.



(c) Resnični izhod modela, imenovan tudi spektrogram iz MIDI datoteke. Na njem se vidi, kaj je dejansko zaigrano.

Slika 4.4: Primer izhoda modela za posnetek ENSTDkAm/MUS/MAPS_MUS-schub_d760_3_ENSTDkAm.wav.
Na zgornji sliki so eden zraven drugega zloženi izhodni vektorji modela. Na sredinski sliki je isti izhod binariziran s pragom 0,5. Tako predelani izhod se nato primerja z resničnim spektrogramom, narejenim iz pripadajoče MIDI datoteke, ki je na spodnji sliki.

zacije za nevronske mreže. Pri vzvratnem razširjanju napake z Dropoutom se naključne povezave ne upoštevajo, kar omogoča, da se model ne prilagodi preveč učnim podatkom.

Pri opisovanju arhitektur smo zaradi preglednosti izpustili Dropout in MaxPool plasti ter izhodno plast. Vsi modeli imajo med vsemi plastmi Dropout plasti z verjetnostjo neupoštevanja 10 %. Za vsako konvolucijsko plastjo je MaxPool plast s filtrom višine 2 in širine 1. Izhodna plast modelov je vedno sestavljena iz 88 nevronov in ima logistično aktivacijsko funkcijo, medtem ko imajo vse druge plasti ReLU aktivacijsko funkcijo. Pri učenju smo uporabili Adam optimizator [13] s stopnjo učenja 0,0006 in binarno entropijsko funkcijo izgube (angl. Binary Cross–entropy loss funciton). Pri modelih smo merili preciznost in priklic. V začetnih poskusih smo opazili, da je priklic vedno veliko nižji od preciznosti, zato smo nastavili priklic kot ciljno metriko pri učenju. Vsi modeli so se najprej učili 10 prehodov, nato pa so se učili do 200 prehodov, vendar se je učenje prekinilo, če se priklic modela ni izboljšal v 10 prehodih.

Zgradili smo modele štirih različnih arhitektur, ki se primarno razlikujejo po številu in vrsti skritih plasti. Arhitektura A ima eno konvolucijsko plast, arhitektura B ima dve konvolucijski plasti, arhitektura C ima eno konvolucijsko plast in eno polno povezano plast in arhitektura D ima dve konvolucijski plasti in eno polno povezano plast. Za vsako arhitekturo smo naredili več različnih modelov, ki so imeli različno število filtrov in nevronov v plasteh.

Velikosti posameznih plasti so zraven imena modelov in njihovega števila parametrov za arhitekture A, B, C in D predstavljene v tabeli 4.1. Pri tem so konvolucijske plasti označene s "Conv n", kjer je n številka plasti, in imajo obliko "štFiltrov $\times$ (višinaFiltra, širinaFiltra)", kjer je štFiltrov število filtrov v plasti, (višinaFiltra, širinaFiltra) pa zapis velikosti filtra. Polno povezane plasti so označene z "Dense n", kjer je n številka plasti, in imajo podano samo število nevronov v plasti.

Velikosti posameznih plasti arhitektur smo izbirali tako, da grajeni modeli pokrivajo čim večji razpon števila parametrov.

Oznaka	Conv 1	Conv 2	Dense 3	Št. parametrov
A1	$8 \times (24, 3)$	–	–	178.080
A2	$16 \times (24, 3)$	–	–	356.072
A3	$24 \times (24, 3)$	–	–	534.064
B1	$8 \times (24, 3)$	$16 \times (12, 3)$	–	55.984
B2	$16 \times (24, 3)$	$32 \times (12, 3)$	–	121.096
B2	$32 \times (24, 3)$	$64 \times (12, 3)$	–	278.968
C1	$8 \times (24, 3)$	–	256	1.056.232
C2	$16 \times (24, 3)$	–	384	2.358.448
C3	$32 \times (24, 3)$	–	384	3.133.176
C4	$32 \times (24, 3)$	–	512	4.176.760
D1	$24 \times (24, 3)$	$48 \times (12, 3)$	384	741.088
D2	$32 \times (24, 3)$	$64 \times (12, 3)$	512	1.301.432
D3	$40 \times (24, 3)$	$96 \times (12, 3)$	512	1.956.384

Tabela 4.1: Hiperparametri in oblike posameznih arhitektur. Arhitektura A ima eno konvolucijsko plast, arhitektura B ima dve konvolucijske plasti, arhitektura C ima eno konvolucijsko plast in eno polno povezano plast in arhitektura D ima dve konvolucijski plasti in eno polno povezano plast.

Poglavje 5

Rezultati

Vsako izmed opisanih arhitektur A, B, C in D smo učili posebej na standardiziranih in posebej na logaritemsko kompresiranih spektrogramih. Pri tem smo za vsako arhitekturo naučili pet modelov, da smo dobili bolj natančne rezultate o metrikah posamezne arhitekture. Vse skupaj je bilo naučenih 130 modelov. Rezultati so predstavljeni v tabeli 5.1, kjer posamezne številke predstavljajo povprečno metriko petih učenih modelov. Rezultati modelov, učenih na standardiziranih spektrogramih, so v stolpcih, označenih s stan (v nadaljevanju: stan modeli), rezultati modelov, učenih na logaritemsko kompresiranih spektrogramih, pa so v stolpcih, označeni z log (v nadaljevanju: log modeli).

Med rezultati se vidi, da imajo log modeli višjo preciznost, priklic in F vrednost. Pri tem je preciznost log modelov v povprečju višja za 0,0454, priklic za 0,0962 in F vrednost za 0,0728 od stan modelov. Stan modeli imajo manjšo napako, ta je v povprečju manjša za 0,00199 od napake log modelov.

Razlike med arhitekturami so minimalne, a prisotne. Najbolj uspešni so modeli arhitekture D s povprečno F vrednostjo 0,696. Sledijo jim modeli arhitekture B s povprečno F vrednostjo 0,687. Tretji najuspešnejši modeli pripadajo arhitekturi A s povprečno F vrednostjo 0,670. Najslabši so modeli arhitekture C, ki imajo povprečno F vrednost 0,665.

	napaka		preciznost		priklic		F vrednost	
oznaka	stan	log	stan	log	stan	log	stan	log
A1	0,0809	0,0813	0,688	0,759	0,581	0,672	0,629	0,713
A2	0,0875	0,0857	0,673	0,724	0,600	0,682	0,634	0,702
A3	0,0937	0,0842	0,673	0,746	0,599	0,675	0,634	0,709
B1	0,0694	0,0781	0,708	**0,782**	0,590	0,657	0,644	0,714
B2	**0,0688**	0,0774	**0,723**	0,763	0,602	0,690	0,657	0,725
B3	0,0690	0,0785	0,713	0,754	**0,614**	0,699	0,659	0,725
C1	0,0939	0,0864	0,672	0,713	0,592	0,701	0,630	0,707
C2	0,106	0,0941	0,666	0,683	0,596	0,713	0,629	0,697
C3	0,108	0,0919	0,668	0,703	0,593	0,705	0,628	0,704
C4	0,118	0,0989	0,654	0,683	0,598	**0,714**	0,625	0,698
D1	0,0701	**0,0771**	**0,723**	0,756	0,609	0,709	**0,661**	0,732
D2	0,0738	0,0781	0,712	0,754	0,609	0,713	0,657	**0,733**
D3	0,0771	0,0797	0,711	0,755	0,612	0,713	0,657	**0,733**

Tabela 5.1: Povprečni rezultati modelov posameznih arhitektur. Stan modeli so v vseh primerih slabši od log modelov. S **krepko** so označeni najboljši rezultati modelov po posameznih metrikah.

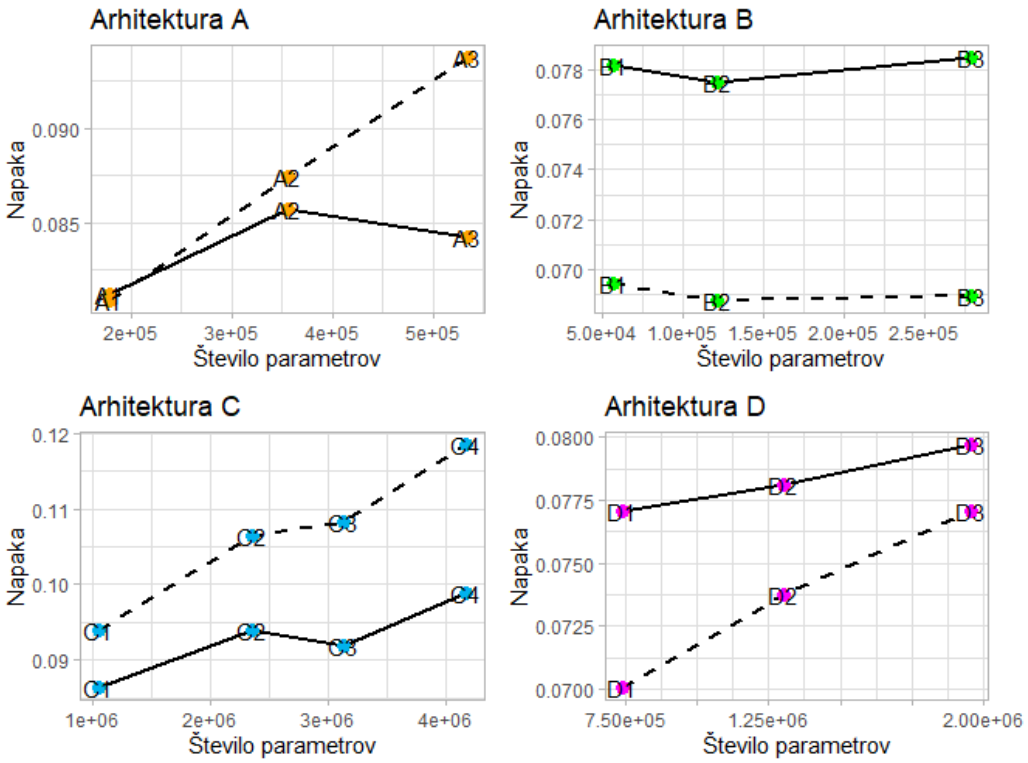
Med vsemi modeli sta arhitekturi B in D dosegali najnižje napake na testnih skladbah. Arhitektura B je imela tudi najvišjo preciznost izmed vseh arhitektur. Pri priklicu sta najboljši rezultat dosegli arhitekturi B in C. Najvišjo F vrednost je dosegla arhitektura D.

Razlike med modeli posameznih arhitektur so grafično prikazane in predstavljene na naslednjih slikah, kjer so s polno črto označeni log modeli in s črtkano črto stan modeli. Slika 5.1 prikazuje napake na testnih skladbah. Pri vseh arhitekturah napaka z velikostjo modela narašča. Pri arhitekturah B in D imajo stan modeli nižjo napako kot log modeli. Slika 5.2 prikazuje preciznost. Pri vseh arhitekturah preciznost pada z velikostjo, stan modeli pa imajo vedno slabšo preciznost kot log modeli. Slika 5.3 prikazuje priklic, ki narašča z velikostjo modela. Tudi tu so log modeli vedno boljši od stan modelov. Slika 5.4 prikazuje F vrednost, za katero izgleda, da pri arhitekturah A in C pada in pri arhitekturah B in D raste s številom parametrov. Log modeli imajo vedno višjo F vrednost kot stan modeli.

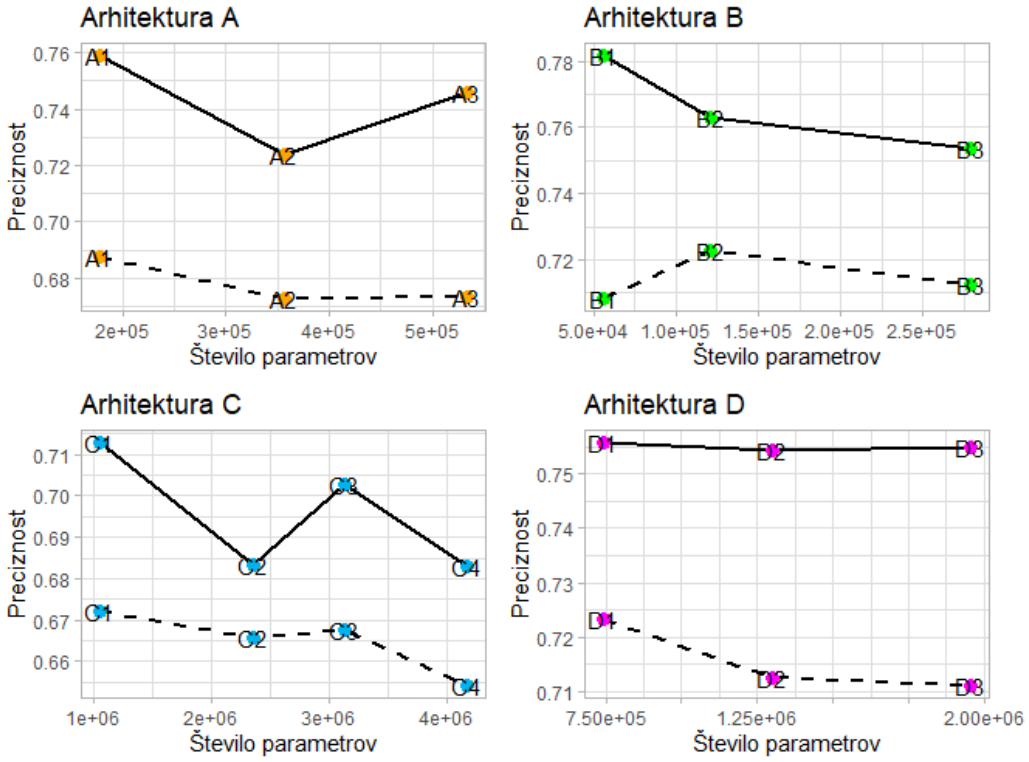
Pri učenju modelov smo merili tudi čas prehodov. V tabeli 5.2 so predstavljeni časi povprečnega prehoda za posamezne modele in arhitekture. Modeli so bili učeni na grafični kartici NVIDIA Quadro RTX 5000. Vsi časi so v sekundah.

Arhitektura A je imela najkrajše prehode, kar je smiselno, saj ima najmanjše modele. Arhitektura D ima najdaljše prehode pri učenju. Arhitektura C ima dolžino prehodov bližje arhitekturi A kot D, kar nam pove, da čas učenja ni odvisen samo od števila parametrov, ampak tudi od oblike modela. Med posameznimi arhitekturami pa so rezultati pričakovani. Pri isti arhitekturi se pri večjem številu parametrov čas enega prehoda podaljša.

Rezultate naših najboljših modelov posameznih arhitektur lahko primerjamo z modeli drugih raziskovalcev, ki so veljali za State-of-the-Art ali dobre modele s področja okvirne avtomatske transkripcije glasbe. Vsi primerjani modeli so bili učeni na podatkovni zbirki MAPS in so imeli enako razdelitev skladb na učne in testne. Za primerjavo smo vzeli model ConvNet, ki so ga predstavili Sigtia in ostali v [25] leta 2016, model AllConv, ki so ga predsta-



Slika 5.1: Prikaz funkcije napake po modelih posameznih arhitektur. S polno črto so označeni log modeli in s črtkano stan modeli. Najvišjo napako ima arhitektura C. V večini arhitektur napaka modela narašča z njegovo velikostjo.

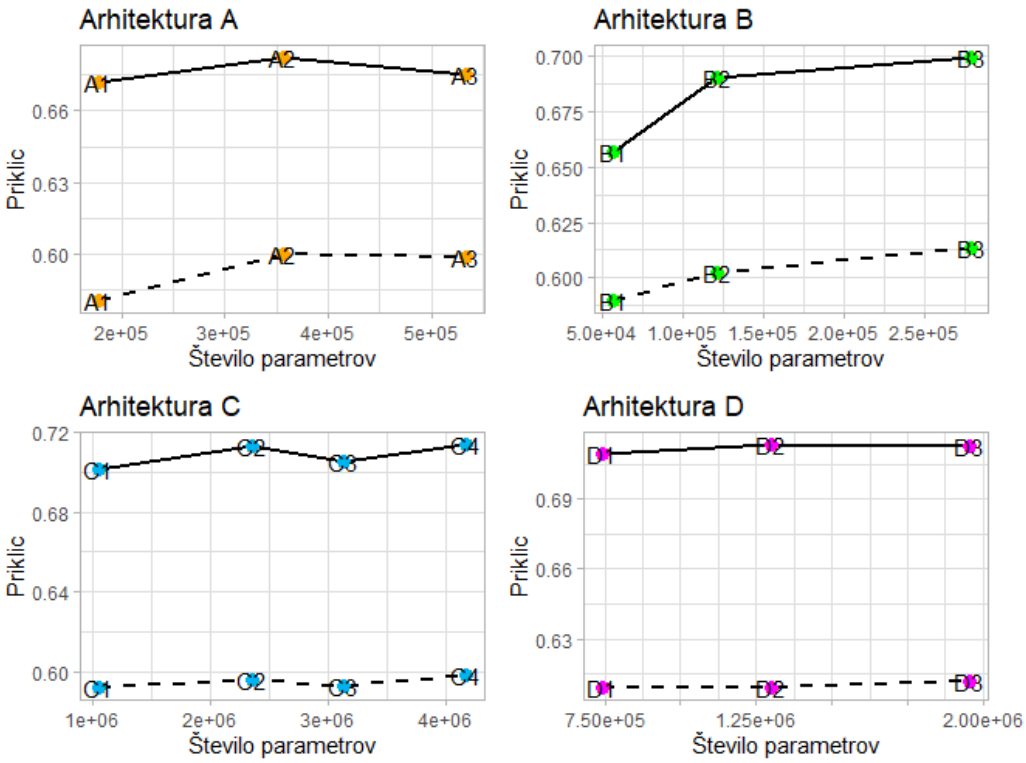


Slika 5.2: Prikaz preciznosti po modelih posameznih arhitektur. V splošnem preciznost pada z velikostjo modela.

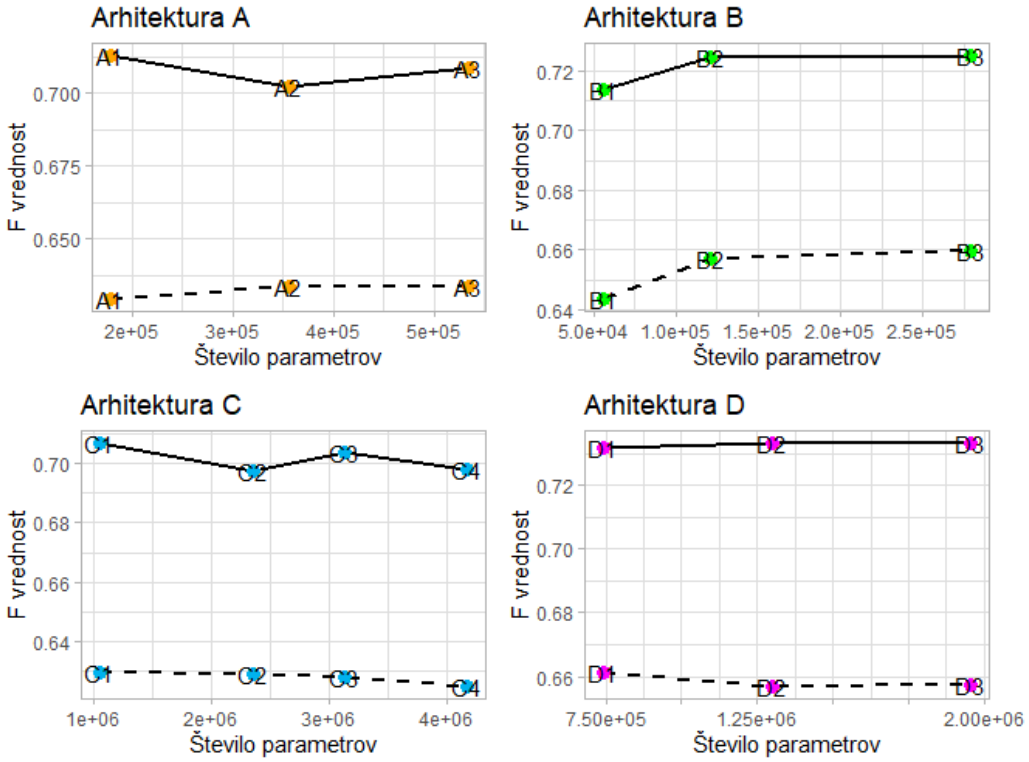
oznaka	stan	log
A1	111.50	110.71
A2	127.19	115.14
A3	127.75	123.83
B1	136.31	133.30
B2	136.20	136.51
B3	190.40	181.23

oznaka	stan	log
C1	127.66	124.64
C2	136.22	143.73
C3	152.08	163.16
C4	159.85	170.04
D1	166.37	161.87
D2	189.94	200.54
D3	250.31	264.48

Tabela 5.2: Povprečni časi enega prehoda pri učenju. Vsi časi so v sekundah [s]. Najdaljši prehod ima arhitektura D, ki ima dve konvolucijski plasti. Čeprav ima arhitektura C največ parametrov, nima najdaljšega časa prehoda.



Slika 5.3: Prikaz priklica po modelih posameznih arhitektur. Priklic narašča z velikostjo modela, kar je po vsej verjetnosti posledica tega, da je priklic ciljna spremenljivka pri učenju.



Slika 5.4: Prikaz F vrednosti po modelih posameznih arhitektur. Log modeli so konstantno boljši od stan modelov za približno 0,05.

vili Kelz in ostali leta 2016 v [12], model OnsetAndFrames, ki so ga naredili Hawthorne in ostali v [8] leta 2018, in model ADSRNet, ki so ga predstavili Kelz in ostali v [11] leta 2019.

Model ConvNet ima dve konvolucijski, dve polno povezani plasti in izhodno plast. Skupno ima 1.789.438 parametrov. Model AllConv ima 7 konvolucijskih plasti in izhodno plast. Kljub številčnim plastem ima samo 284.544 parametrov. Model OnsetAndFrames pa je mnogo kompleksnejši, saj vsebuje detektor za začetke not in okvirni detektor. Njegova arhitektura je podrobneje predstavljena v njegovem članku. Uradna številka parametrov ni znana, Kwon in ostali v [14] pa ocenjujejo, da ima model med 18.3 in 23.5 milijonov parametrov. Model ADSRNet je sestavljen iz šestih konvolucijskih in treh izhodnih plasti. Izhodne plasti služijo za napoved začetka, sredine in konca določene note. Model ima 326.394 parametrov.

Rezultate smo primerjali v tabeli 5.3. Vsi naši modeli so boljši od modela AllConv. Po uspešnosti mu je najbližji C4, a ima ta 14-krat več parametrov. Modela B2 in D2 sta boljša od ConvNet, pri tem ima B2 14-krat manj parametrov. D2 in ConvNet sta si po številu parametrov bližje, a jih ima D2 tretjino manj. ADSRNet in OnsetAndFrames pa sta mnogo boljša. OnsetAndFrames dosega skupno najboljše rezultate, kar je po vsej verjetnosti posledica njegove velikosti in kompleksnosti. ADSRNet je veliko manjši in po parametrih primerljiv z modelom A2, a ima veliko več konvolucijskih plasti, ki zmanjšajo število parametrov in so bolj primerne za učenje iz spektrogramov kot polno povezane plasti.

oznaka	Št parametrov	preciznost	priklic	F vrednost
A1	178.080	0,759	0,672	0,713
B2	121.096	0,763	0,690	0,725
C4	4.176.760	0,683	0,714	0,698
D2	1.301.432	0,755	0,713	0,733
ConvNet [25]	1.789.438	0.720	0,733	0,722
AllConv [12]	284.544	0,765	0,635	0,694
ADSRNet [11]	326.394	0,907	0,679	0,772
OnsetAndFrames [8]	18,3-23,5 mio.	0,885	0,709	0,783

Tabela 5.3: Log modeli primerjani z modeli drugih avtorjev. Modeli so boljši od modela AllConv iz leta 2016 in primerljivi z modelom ConvNet iz leta 2016, ki ima od ostalih najmanj kompleksno arhitekturo. ADSRNet in OnsetAndFrames dosegata veliko višje rezultate.

Poglavje 6

Zaključek

Iz grafov rezultatov vidimo, da preciznost pada in priklic narašča z velikostjo modela. To je po vsej verjetnosti posledica dejstva, da je priklic glavna metrika pri učenju. Večji modeli lahko dosegajo boljše rezultate in na grafih priklica je to tudi prikazano. Vendar splošno uspešnost modela merimo z F vrednostjo, ki pa je nismo mogli imeti za ciljno, saj bi isto F vrednost lahko model dosegel z različnim naborom preciznosti in priklica.

Iz rezultatov vidimo, da imajo višje F vrednosti arhitekture z več konvolucijskimi plastmi. Taki arhitekturi sta B in D. D ima za razliko od arhitekture B še eno dodatno navadno plast, kar ji očitno da prednost pred arhitekturo B. Stvar se ravno obrne pri arhitekturama A in C. Obe imata samo eno konvolucijsko plast, arhitektura C ima za njo še eno navadno plast. Po številu parametrov ima arhitektura C veliko več parametrov kot arhitektura A, vendar je arhitektura A boljša od arhitekture C. To si lahko razlagamo tako, da je arhitektura C prevelika in ni bila dovolj časa učena ali da je arhitektura A ravno dovolj majhna in je sposobna bolj generalizirati na nevidenih podatkih.

Število parametrov pri različnih arhitekturah tako ne pripomore k boljšim rezultatom. Arhitekturi B in D sta bili najbolj uspešni in sta imeli razliko F vrednosti le 0.009. Najbolj uspešen model arhitekture B (B3) je imel 7-krat manj parametrov kot najbolj uspešen model arhitekture D (D3). Arhitektura C je imela povprečno največ parametrov in bila najslabša izmed vseh.

Največji model arhitekture C (C4) je imel 74-krat več parametrov kot najmanjši model arhitekture B (B1), dosegel pa je slabšo F vrednost. Kljub temu je imel model C4 najvišji priklic izmed vseh, a nižjo preciznost in F vrednost, kar ga je uvrstilo med slabše modele.

Zanimivi so tudi rezultati znotraj arhitektur samih. Pričakovano je, da je večji model boljši, a v našem primeru temu ni ravno tako. Pri stan modelih to velja pri arhitekturah B in D, za A in C pa to ne velja. Pri log modelih so večji modeli boljši pri arhitekturah A in C, kar je ravno obratno od stan modelov. Pri takem pregledu žal ne moremo reči nič konkretnega, saj bi potrebovali veliko več modelov iste arhitekture za boljše in natančnejše rezultate. Iz samo treh ali štirih različnih modelov iste arhitekture tako ne moremo potrditi, da z istim številom konvolucijskih plasti in različnim številom parametrov nujno dobimo boljše rezultate. Lahko pa iz primerjav arhitektur med sabo ugotovimo, da so konvolucijske plasti ključnega pomena in tako arhitekture z več konvolucijskimi plastmi dosegajo boljše rezultate.

Ugotovitev o uspešnosti konvolucijskih plasti podpira tudi primerjava z drugimi modeli. AllConv je po parametrih primerljiv z našimi manjšimi modeli in dosega primerljive, a malo slabše rezultate. ADSRNet je manjši od veliko naših modelov in dosega veliko višje rezultate. Malo slabši je od modela OnsetAndFrames, ki ima kar 60-krat več parametrov. ADSRNet s svojo arhitekturo ni bistveno kompleksnejši od naših modelov, ima pa prednost, da se hkrati uči začetke, sredine in konce not. Prav tako ADSRNet modelira note po principu ADSR ovojnice (angl. Attack, Decay, Sustain, and Release envelope), ki opisuje obnašanje note skozi čas. Ta razlika mu omogoča bistveno boljše rezultate z minimalnimi in pametnimi spremembami v arhitekturi. OnsetAndFrames je zgrajen iz dveh modelov in poleg konvolucijskih plasti uporablja še LSTM plasti, ki so narejene posebej za podatke, povezane s časom. Edina slabost OnsetAndFrames modela je njegova velikost. Kljub zelo dobrim rezultatom ni bistveno boljši od ADSRNeta, kar nam da misliti, da velikost modela ni vse. Velik pomen imata arhitektura in zasnova modela. ADSRNet je dokaz, da lahko z relativno majhnim modelom naredimo

uspešno transkripcijo.

Pri tem pa konvolucijske plasti veliko bolj podaljšajo čas učenja kot navadne plasti. Kljub veliko večjim navadnim plastem pri arhitekturi C je imela arhitektura B z eno konvolucijsko plastjo enake ali daljše prehode pri učenju, arhitektura D z dvema konvolucijskima plastema pa še daljše. Pri tem so bile F vrednosti slednjih dveh le za $0,03$ višje od arhitekture C. Model D3 je imel 2-krat daljši čas prehoda kot model A2, a je dosegel le $0,03$ višjo F vrednost. Arhitektura D je res uspešnejša od arhitekture A, a obstaja vprašanje, ali je 2-krat daljši čas prehodov in posledično 2-krat daljše učenje vredno minimalnih izboljšav.

Ugotovili smo, da število parametrov ni povezano z uspešnostjo modela. Prav tako ni povezano z dolžino učenja modela. Izkazalo se je, da je uspešnost povezana s številom konvolucijskih plasti. Prav tako enostavni in manjši modeli ne dosegajo bistveno slabših rezultatov, da pa se jih izboljšati z rahlo kompleksnejšimi arhitekturami in bolj naprednimi pristopi. V nadaljevanju bi lahko testirali nevronske mreže z večjim številom konvolucijskih plasti, kar bi poslabšalo čas prehoda pri učenju in s tem podaljšalo učenje. Prav tako bi se splačalo raziskati enostavnejše arhitekture s konvolucijskimi plastmi in s plastmi s povratno zanko, ki so bile v preteklosti uspešno uporabljene, a na velikih modelih. Veliko uspešnih modelov prav tako uporablja različne detektorje za zaznavo začetkov not. S temi si lahko modeli veliko pomagajo, saj note skozi čas nimajo konstantne amplitude. Note, zaigrane na klavirju, imajo močan začetek, a potem počasi izzvenijo. To znanje izkorišča ADSR ovojnica, ki je sposobna modelirati tako obnašanje. V prihodnje bi se enostavnejše modele dalo nadgraditi z detektorjem začetkov in z ADSR ovojnico, kar bi po vsej verjetnosti močno izboljšalo rezultate.

Literatura

[1] Emmanouil Benetos in sod. "Automatic Music Transcription: An Overview". V: *IEEE Signal Processing Magazine* 36.1 (2019), str. 20–30. DOI: 10.1109/MSP.2018.2869928.

[2] Sebastian Böck in Markus Schedl. "Polyphonic piano note transcription with recurrent neural networks". V: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2012, str. 121–124. DOI: 10.1109/ICASSP.2012.6287832.

[3] Judith C. Brown. "Calculation of a constant Q spectral transform". V: *The Journal of the Acoustical Society of America* 89.1 (1991), str. 425–434. DOI: 10.1121/1.400476. eprint: https://doi.org/10.1121/1.400476. URL: https://doi.org/10.1121/1.400476.

[4] Keunwoo Choi in sod. *A Comparison of Audio Signal Preprocessing Methods for Deep Neural Networks on Music Tagging*. 2021. arXiv: 1709.01922 [cs.SD].

[5] Karin Dressler. "Extraction of the melody pitch contour from polyphonic audio". V: *Proc. 6th International Conference on Music Information Retrieval*. Zv. 110. Citeseer. 2005. DOI: 10.1.1.122.7773.

[6] Valentin Emiya, Roland Badeau in Bertrand David. "Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle". V: *Audio, Speech, and Language Processing, IEEE Transactions on* 18 (sep. 2010), str. 1643–1654. DOI: 10.1109/TASL.2009.2038819.

[7] Michael Jerome Hawley. "Structure out of sound". Doktorska disertacija. Massachusetts Institute of Technology, 1993. URL: http://hdl.handle.net/1721.1/29068 (pridobljeno 8. 8. 2021).

[8] Curtis Hawthorne in sod. *Onsets and Frames: Dual-Objective Piano Transcription*. 2018. arXiv: 1710.11153 [cs.SD].

[9] Eric J. Humphrey in Juan P. Bello. "Rethinking Automatic Chord Recognition with Convolutional Neural Networks". V: *2012 11th International Conference on Machine Learning and Applications*. Zv. 2. 2012, str. 357–362. DOI: 10.1109/ICMLA.2012.220.

[10] Haruhiro Katayose in Seiji Inokuchi. "The Kansei Music System". V: *Computer Music Journal* 13.4 (1989), str. 72–77. ISSN: 01489267, 15315169. URL: http://www.jstor.org/stable/3679555.

[11] Rainer Kelz, Sebastian Böck in Gerhard Widmer. "Deep polyphonic ADSR piano note transcription". V: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, str. 246–250.

[12] Rainer Kelz in sod. *On the Potential of Simple Framewise Approaches to Piano Transcription*. 2016. arXiv: 1612.05153 [cs.SD].

[13] Diederik P. Kingma in Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980.

[14] Taegyun Kwon, Dasaem Jeong in Juhan Nam. *Polyphonic Piano Transcription Using Autoregressive Multi-State Note Model*. 2020. arXiv: 2010.01104 [eess.AS].

[15] D. Lee in H. Seung. "Learning the parts of objects by non-negative matrix factorization". V: *Nature* 401 (okt. 1999), str. 788–791. DOI: 10.1038/44565.

[16] Matija Marolt. "A connectionist approach to automatic transcription of polyphonic piano music". V: *IEEE Transactions on Multimedia* 6.3 (2004), str. 439–449.

[17] Martín Abadi in sod. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (pridobljeno 4. 8. 2021).

[18] Brian McFee in sod. "librosa: Audio and Music Signal Analysis in Python". V: Proceedings of the 14th python in science conference, jan. 2015, str. 18–24. DOI: 10.25080/Majora-7b98e3ed-003.

[19] *MIREX Wiki*. Jun. 2021. URL: https://www.music-ir.org/mirex/wiki/MIREX_HOME (pridobljeno 4. 8. 2021).

[20] Brian C. J. Moore. *An Introduction to the Psychology of Hearing*. Brill, 2012.

[21] James A. Moorer. "On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer". Doktorska disertacija. Stanford, CA: Stanford University, 1975. URL: https://ccrma.stanford.edu/files/papers/stanm3.pdf (pridobljeno 8. 8. 2021).

[22] Graham Poliner in sod. "Melody Transcription From Music Audio: Approaches and Evaluation". V: *Audio, Speech, and Language Processing, IEEE Transactions on* 15 (jun. 2007), str. 1247–1256. DOI: 10.1109/TASL.2006.889797.

[23] Colin Raffel in Daniel P. W. Ellis. "Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi". V: *Proceedings of the 15th International Conference on Music Information Retrieval Late Breaking and Demo Papers* (2014), str. 84–93.

[24] Jan Schlüter in Sebastian Böck. "Improved musical onset detection with Convolutional Neural Networks". V: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, str. 6979–6983. DOI: 10.1109/ICASSP.2014.6854953.

[25] Siddharth Sigtia, Emmanouil Benetos in Simon Dixon. "An End-to-End Neural Network for Polyphonic Piano Music Transcription". V: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.5 (2016), str. 927–939. DOI: 10.1109/TASLP.2016.2533858.

[26] Siddharth Sigtia in sod. "A hybrid recurrent neural network for music transcription". V: apr. 2015, str. 2061–2065. DOI: 10.1109/ICASSP.2015.7178333.

[27] Paris Smaragdis in Judith Brown. "Non-negative matrix factorization for polyphonic music transcription". V: *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (nov. 2003), str. 177–180. DOI: 10.1109/ASPAA.2003.1285860.

[28] Nitish Srivastava in sod. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". V: *Journal of Machine Learning Research* 15.56 (2014), str. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[29] Li Su in Yi-Hsuan Yang. "Combining Spectral and Temporal Representations for Multipitch Estimation of Polyphonic Music". V: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.10 (2015), str. 1600–1612. DOI: 10.1109/TASLP.2015.2442411.