TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Tambet Uutsalu

Muusika genereerimine närvivõrkude abil

Bakalaureusetöö (9 EAP)

Juhendaja: Sven Aller

Tartu 2019

Muusika genereerimine närvivõrkude abil

Lühikokkuvõte:

Bakalaureusetöö eesmärk oli luua muusikat genereeriv programm, mis suudaks seda teha mitmele heliloojale ning stiilile omaselt, luues kõigepealt taktide põhine struktuur ning selle põhjal üksikud noodid. Töö tulemusena valmis programm, mis võimaldab genereerida muusikat, kasutades seitsmel erineval andmestikul treenitud mudeleid.

Võtmesõnad:

Intellektitehnika, muusika, neurovõrgud

CERCS: P176 Tehisintellekt

Generating music with neural networks

Abstract:

The purpose of this Bachelor's thesis was to create a music generating program that could do it in the style of multiple composers, by first creating the bar based structure and based on that structure generates single notes. The results of this work is a program, that is capable of generating music using models trained on 7 different datasets.

Keywords:

Artificial intelligence, music, artificial neural networks

CERCS: P176 Artificial intelligence

Sisukord

Sissejuhatus

Masinõppe meetodite kasutamine loomingulisteks tegevusteks on koos masinõppe arenguga saanud aina rohkem tähelepanu. Ka muusikalise loomingu jaoks on viimasel ajal välja töötatud palju erinevaid mudeleid, millest mõne poolt genereeritud muusika on võrreldav inimeste omaga [1], kuid enamusel rekurrentsetel närvivõrkudel põhinevatest mudelitest puudub muusikal läbiv teema ja struktuur.

Muusika genereerimiseks on kaks põhilist võimalust: genereerida heli või muusikalist notatsiooni, antud töös vaadatakse viimast, kasutades notatsioonina standardsele noodikirjale vastavaid noodikõrguseid ja -pikkuseid ning andmetena MIDI formaadis faile. Töö eesmärgiks on luua muusikat genereeriv programm, mis suudaks luua muusikat mitmele erinevale heliloojale ning stiilile omaselt. Programmi jaoks luuakse tehisnärvivõrgul põhinev mudel, mis genereerib treeningandmestike põhjal muusikat nii, et kõigepealt luuakse muusikateose taktipõhine struktuur ning seejärel loodud struktuuri põhjal üksikud noodid. Et selle mudeli täpsust ja loomingut saaks võrrelda, luuakse teine mudel, mis genereerib muusikat ennustades noote ainult eelnevate nootide põhjal.

Bakalaureusetöö esimeses osas on ära toodud kasutatud terminite seletused, kasutatavate tehisnärvivõrkudega seonduvate meetodite kirjeldused ning ülevaade olemasolevatest muusikat genereerivatest mudelitest. Teises osas on toodud üldine informatsioon loodava programmi kohta, selgitatud andmete esitusviisi, kirjeldatud loodud rekurrentsetel närvivõrkudel põhinevaid mudeleid ning nende treenimisprotsessi, vaadatud ja võrreldud mudelite poolt saavutatud tulemusi ning toodud näiteid mudelite poolt genereeritud muusika kohta. Kolmandas osas on toodud programmi paigaldus- ja kasutusjuhend, millele järgneb kokkuvõte. Kokkuvõttele järgnevad lisad: lisas 1 on toodud detailne mudeli struktuur, lisas 2 on toodud kasutatavad noodipikkused ning lisas 3 on viited genereeritud muusika näidetele.

1. Muusika ja tehisnärvivõrgud

Selles peatükis on toodud MIDI formaadi kirjeldus, antud töö jaoks kasutatud tehisnärvivõrkudega seonduvate meetodite kirjeldused ning näiteid erinevate tehisnärvivõrkude abil loodud muusikat genereerivate mudelite kohta.

1.1 MIDI

MIDI ehk muusika instrumendi digitaalne liides (ingl *Musical Instrument Digital Interface*) on failiformaat, mis on mõeldud ajaliselt märgistatud MIDI andmete edastamiseks programmide vahel ühes või erinevates arvutites[1].

MIDI failid koosnevad paralleelsetest radadest, mis omakorda sisaldavad sõnumeid. Iga sõnumi jaoks on määratud möödunud aeg eelmisest sõnumist[2].

Antud töö jaoks on MIDI failidest kasutusel järgmised elemendid:

- noodikõrgus - määratud *note-on* sõnumis väärtusega vahemikus 0-127;
- noodi valjus - määratud *note-on* sõnumis väärtusega vahemikus 0-127;
- noodipikkus - määratud sama noodi kohta käivate *note-on* ja *note-off*[3] sõnumite vahele jäävate sõnumite aegade summana. Aja ühikuks on MIDI-tiks (ingl *MIDI tick*);
- tiksude arv ühes löögis – määratud MIDI faili päises;
- taktimõõt – määratud *time-signature* sõnumis.

MIDI formaadi noodipikkused pole piiratud ning võivad erineda ka ainult ühe tiksu poolest, see teeb noodikirjast vaste otsimise keeruliseks.

1.2 Tehisnärvivõrgud

RNN

Rekurrentne tehisnärvivõrk (RNN) on tehisnärvivõrk, mis sisaldab vähemalt ühte tagasisideahelat[4] ja suudab käsitleda erinevate pikkustega sisendandmete järjendeid, kasutades selleks rekurrentset peidetud olekut (ingl *recurrent hidden state*), mille aktivatsioon igal ajasammul sõltub eelmise ajasammu omast (valem 1) [2] ehk sisendjärjendi $X = (x_1, x_2, .., x_T)$ korral:

$$h_t = \begin{cases} 0, & t = 0 \\ \varphi\,(h_{t-1}, x_t), & t > 0 \end{cases} \qquad (1)$$

kus h_t on peidetud olek ajahetkel t ja φ on mittelineaarne funktsioon.

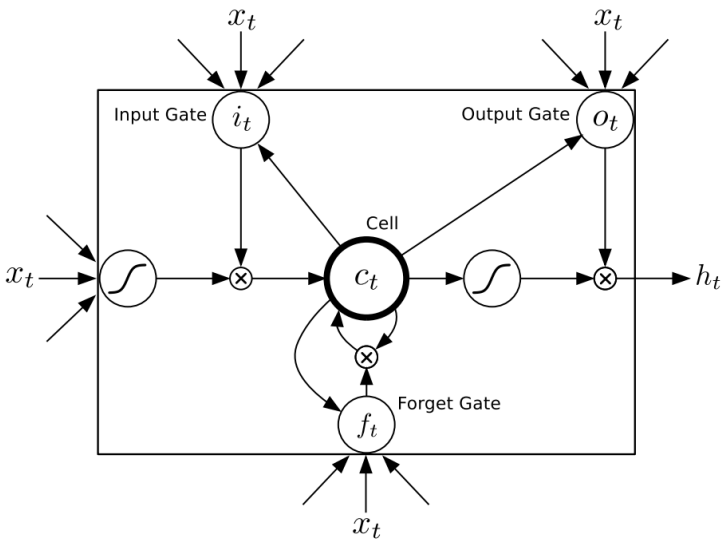
RNN-de treenimine pikaajaliste sõltuvuste tajumiseks on raske, kuna esineb haihtuva gradiendi probleem (ingl *vanishing gradient problem*) [3].

LSTM

LSTM ehk pikk lühiajaline mälu (ingl *Long Short-Term Memory)* [4] on rekurrentne närvivõrk (RNN), mis koosneb LSTM ühikutest (ingl *LSTM unit*). LSTM ühik koosneb mälust (joonisel 2 *Cell*), sisendväravast (*Input Gate*), väljundväravast (*Output Gate*) ja unustamise väravast (*Forget Gate*).

---

[1] http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html
[2] http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html
[3] Noodi lõppu tähistab ka *note-on* sõnum, mille helivalidus (ingl *velocity*) on 0.
[4] http://eki.ee/dict/its/index.cgi?F=num&C10=1&Q=2899



Joonis 1. LSTM ühik [5]

LSTM ühiku olek h_t sõltub ajahetkel t sisendist x_t ning eelmisest olekust h_{t-1} ja eelmisest mälu olekust c_{t-1}, unustamise ja sisendväravad määravad vastavalt kui palju eelmist mälu c_{t-1} unustada ja uut lisada (kujutatud joonisel 2) [5]. LSTM eelis tavalise RNN ees avaldub LSTM võimes leida ja ära kasutada seoseid pikema aja tagant ning LSTM treenimisel ei teki haihtuva gradiendi probleemi [2]. LSTM neurovõrgud sobivad hästi ajaliselt järjestatud andmete põhjal ennustuste tegemiseks ning on kasutatud mitmed kordi ka muusika genereerimiseks [6,7].

Dropout

Alasobitatuks (ingl *underfitting*) nimetatakse mudelit, kui mudel on liiga üldine ning ei ole täpne treeningandmestiku peal ega ka enne nägemata andmestike peal.

Ülesobitatuks (ingl *overfitting*) nimetatakse mudelit, mis ei üldista piisavalt ning jäädvustab olematuid mürast tulenevaid seoseid. Ülesobitatud mudel on täpne treeningandmete kohta, aga ebatäpne uute andmete peal [8].

Dropout on regulariseerimismeetod ülesobitamise vähendamiseks tehisnärvivõrkudes. See töötab jättes võrgust välja juhuslike neuroneid (koos nende ühendustega) eelnevalt määratud tõenäosuse p järgi, vähendades sellega võimalike seoste hulka ning suurendades üldistusvõimet [9]. *Dropout* on kasutusel tehisnärvivõrgu treenimisel, kus iga treeningjuhu juures väljajäetud neuronid muutuvad; testimisel kasutatakse kõiki neuroneid, aga neuronite kaalud korrutatakse läbi p -ga, et eeldatavad väljundid oleksid samaväärsed [9].

Siin töös kasutatud Kerasee *dropout* kiht teeb etteantud r osa sisendühikutest 0-ks ning kõik kõik sisendühikute kaalud jagatakse läbi r-ga, et testimisel ei peaks korrutama[5].

1.3 Tehisnärvivõrkudel põhinevad muusikat genereerivad mudelid

Muusika genereerimiseks on kaks põhilist võimalust: genereerida noodikirja või heli. Siin peatükis on toodud 5 näidet noodikirja genereerivate mudelite kohta ning 1 näide heli genereeriva mudeli kohta.

[5] https://github.com/keras-team/keras/blob/master/keras/layers/core.py#L81

CONCERT

Süvaõppe meetodeid on kasutatud muusika genereerimiseks juba pikka aega. 1994. aastal loodi rekurrentne närvivõrk CONCERT muusika automaatseks loomiseks [10]. Muusika andmete esitluseks kasutati esitlusviisi, mis koosnes sündmustest ja iga sündmus koosnes noodikõrgusest, noodipikkusest ning noodiga koos kõlavast akordist. CONCERT ennustas etteantud sündmuste jada põhjal järgmist sündmust ehk noodikõrgust, -pikkust ja samal ajal mängivat akordi. Autori sõnul puudus päris muusika peal treenitud mudeli poolt genereeritud muusikal[6] struktuur ning rütmiline organiseeritus [10].

LSTM-il põhinev mudel

2002. aastal loodi LSTM-il põhinev mudel, mis suutis ära õppida bluusi žanris lihtsa muusika akordide struktuuri ja suutis luua ka akordidega kokku käiva meloodia. Andmete esitluseks kasutati esitlusviisi, mis koosnes ajasammudest (ingl *time-step*), mille kohta oli kirjas, mis noot meloodias mängib ja mis noodid akordis mängivad. Noodid valiti pentatoonilisest helireast ja treenimiseks kasutati ainult ühte akordijada. Kuigi kasutatavad noodid ja akordid olid piiritletud, oli loodav muusika[7] autori arvates kohati küllakti meeldiv [6].

Performance RNN

2017. aastal loodi Perfomance RNN – LSTM-il põhinev rekurrentne närvivõrk, mille eesmärgiks oli polüfoonilise klaverimuusika genereerimine, nii et loodud muusika oleks liikuv ja ilmekas [7]. Selle saavutamiseks treeniti mudelit klaveriesituste MIDI salvestiste peal. Sellel mudelil olid võimalikud kõik noodipikkused ja nootide kombinatsioonid, samuti genereeris see mudel tempo- ja helivaljuse muutuseid. Muusika andmete esitamiseks kasutati esitlusviisi, mis koosnes sellistest sündmustest:

- 128 nooti alustavat sündmust (iga võimaliku MIDI noodikõrguse kohta üks sündmus);
- 128 nooti lõpetavat sündmust;
- 100 aega edasi viivat sündmust, mis vastasid aegadele 10ms kuni 1s;
- 32 helivaljust muutvat sündmust, mis määrasid järgnevate nootide helivaljused.

Nii ennustas see mudel etteantud sündmuste jada põhjal järgmist sündmust. Kuigi mudeli poolt genereeritud muusika[8] on ilmekas ja liikuv ning lühiajalised fraasid on kõrvale meeldivad, puudub fraaside vaheline ühtsus ning pikaajaline muusikaline struktuur [7].

BachProp

2018. aastal loodi LSTM-il põhinev mudel BachProp [1], mille eesmärgiks oli MIDI faile kasutades muusika genereerimine erinevates stiilides vastavalt treeningandmetele. Andmete esitamiseks teisendati MIDI failid kujule, kus teos on nootidest koosnev järjend ja iga noodi kohta on määratud noodikõrgus, noodipikkus ning vahe antud ja eelneva noodi alguste vahel (sama ühik, mis noodipikkusel). Noodipikkused ja viivitused lähendati 21-le levinuimale noodipikkusele.

[6] http://www.cs.colorado.edu/~mozer/Research/Selected%20Publications/music.html
[7] http://www.iro.umontreal.ca/~eckdoug/blues/index.html
[8] https://magenta.tensorflow.org/performance-rnn

Music Transformer

2018. aastal loodi Music Transformer mudel, mis põhineb tähelepanul (ingl *attention*) ning, vastupidiselt eelnevatele siin mainitud mudelitele, ei kasuta see rekurrentset tehisnärvivõrku[11]. Muusikaliste andmete esitusviisina kasutati kahte erinevat: sama mis Performance-RNN ja teiseks ruudustikul põhinevat esitust, kus iga ajasammu kohta on märgitud kõigi noodikõrguste kohta, kas antud noodikõrgus sellel hetkel mängib või ei. „Music Transformer" poolt loodud muusikal[9] on märgatavam pikaajaline struktuur kui eelnevate mudelite puhul ning kui treenimiseks kasutati inimeste poolt mängitud esituste salvestisi, oli muusika ka ilmekas ja liikuv [11].

WaveNet

Kõikide eelnevalt mainitud mudelite jaoks on muusika ehituskivideks noodid, kuid on loodud ka mudeleid, mis genereerivad helilainet. Üks neist on WaveNet [12], mis on mõeldud kõnesünteesi jaoks, kuid treenides mudelit klaverimuusikaga genereeris see ka muusikat[10]. WaveNet lõi muusikat genereerides heli 1 ajasammu kaupa diskreetimissagedusega 16kHz ja kuigi loodud muusika stiil, tempo ning kvaliteet muutusid iga sekundi tagant, olid mudeli poolt loodud heliklipid esteetiliselt meeldivad [12].

[9] https://storage.googleapis.com/music-transformer/index.html
[10] https://deepmind.com/blog/wavenet-generative-model-raw-audio/

2. Muusika genereerimine, kasutades rekurrentseid närvivõrke

Kõik eelmises peatükis tutvustatud muusikalist notatsiooni genereerivad mudelid teevad seda, luues noote või noodi pikkuseid sündmuseid, sellisel lähenemisel peab mudel ise treenimise jooksul õppima muusika suuremat struktuuri, näiteks akordide järgnevust. Samas on võimalik luua mudel just selle jaoks. Selle töö eesmärk ongi luua muusikat genereeriv programm, mis kasutataks eraldi mudelit muusikaliste taktide kohta käivate suuruste ennustamise jaoks, koos teise mudeliga, mis esimesest saadud ennustuste põhjal genereeriks noote. Selle programmi jaoks luuakse kolm mudelit:

1. taktimudel, mis taktide jada põhjal ennustab järgmist takti;
2. noodipõhine mudel, mis ennustab nootide jada põhjal järgmist nooti, loodud selleks võrrelda kombineeritud mudeliga.
3. kombineeritud mudel, mis on samasugune nagu noodipõhine, aga kasutab sisendina lisaks nootidele taktimudeli poolt loodud takte.

Mudelite treenimiseks kasutatakse andmetena MIDI formaadis faile, kuid et andmed esitaksid vaid mudelite treenimiseks olulist informatsiooni, luuakse andmete esitamiseks uus esitusviis, milles on nootide kohta kirjas ainult mudeli poolt ennustatavad suurused.

Seejärel mudeleid treenitakse erinevatel andmestikel ning iga andmestiku kohta salvestatakse eraldi kaalud, et programm oleks võimeline genereerima muusikat mitmes erinevas stiilis.

Valminud programm võimaldab kasutajal genereerida muusikat, kasutades erinevaid mudeleid koos erinevatel andmestikel saadud kaaludega ja neid omavahel kombineerida. Programmi lihtsamaks kasutamiseks luuakse sellele käsurea põhine kasutajaliides.

Käesoleva töö jaoks loodud lähtekood, kasutatud treeningandmed ja muud failid on leitavad antud töö repositooriumist[11]. Kõikide kasutatud tehisnärvivõrkude realiseerimiseks kasutati Kerast [13] koos TensorFlow'ga [14].

2.1 Andmed

Käesoleva töö jaoks kasutatakse treeningandmestikuna MIDI formaadis faile, mis pärinevad piano-midi.de[12] ning bachcentral.com[13] veebilehelt. Siin peatükis kirjeldatakse kasutatavat andmete esitust ning andmete teisendamise protsessi.

Muusikaliste andmete esitus

Kuna MIDI failid sisaldavad peale antud ülesande jaoks vajalike andmete veel palju muud informatsiooni, siis teisendati andmed kujule, mis sisaldaksid vaid mudeli treenimiseks vajalikku ning oleks paremini vastavuses standardse noodikirjaga. Nootide tähistamiseks on kasutatud sarnast esitusviisi nagu kasutati BachProp'i [1] jaoks. Sellise esitusviisiga on muusikateos järjend, mis koosneb ainult noote tähistavatest elementidest ja iga elemendi jaoks on määratud järgmised:

- noodikõrgus – määrab heli sageduse; diskreetne suurus 88 elemendilisest hulgast;
- noodi valjus – täisarvuline väärtus [0-127];
- noodipikkus – näitab, kui kaua noot kestab; diskreetne suurus 12 elemendilisest hulgast;

[11] https://gitlab.com/TambetU/musicgen
[12] http://www.piano-midi.de/
[13] http://www.bachcentral.com

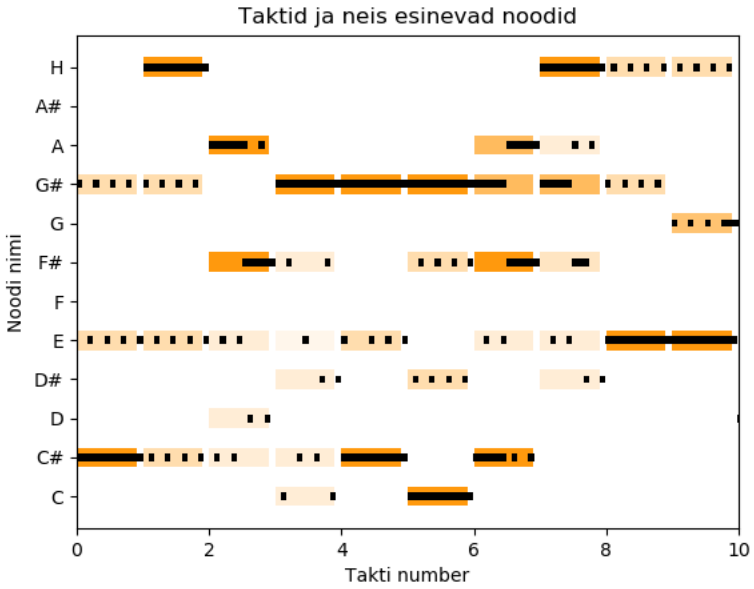
- viivitus – näitab kui kaua peale eelneva noodi mängimist hakkab antud noot mängima, samaväärsed klassid noodipikkusega; diskreetne suurus 12 elemendilisest hulgast.

Võrreldes BachProp esitusviisiga on erinevus selles, et seal kasutati viivituste ja noodipikkustena 21 erinevat väärtust ning 128 noodikõrgust [1].

Muusikateostes leidub üldjuhul üldisem struktuur kui üksikud noodid, näiteks akordide järgnevus. Sellise struktuuri esitamiseks kasutatakse muusikalistele taktidele vastavaid vektoreid. Taktivektor koosneb 12 elemendist, iga element vastab ühele noodikõrgusele oktavis ning näitab kui palju vastav noodikõrgus antud taktis esineb. Elementide väärtused on vahemikus 0-1: 0 puhul nooti ei esine, 1 puhul noot mängib terve takti jooksul (joonis 2).

Andmete teisendamine

MIDI formaadis andmete teisendamiseks ülaltoodud kujule itereeriti üle kõigi MIDI radade ning üle kõigi noodi algust ja lõppu tähistavate sõnumite, viies kokku sellised sõnumid, mis käivad sama noodi kohta ning jättes meelde nootide algusajad. Seejärel sorteeriti nootide järjend algusaegade järgi kasvavalt ning leiti noodipikkused, lahutades noodi lõpuajast algusaja, ja viivitused, lahutades antud noodi algusajast talle eelnenud noodi algusaja. Noodipikkuste ja viivituste teisendamiseks noodikirjale vastavateks, leiti 64-ndik noodi pikkus MIDI tiksudes, kasutades MIDI päisest loetud tiksude arvu ühes löögis ja taktimõõtu. Seejärel võrreldi 64-ndik noodi kordseid uuritava noodipikkusega (täpsed noodipikkused on kirjas lisa 2). Noodipikkuste võrdlemine toimus suurimast väikseimani, kuni uuritav noodipikkus oli suurem või võrdne võrreldava noodipikkusega ehk kui MIDI failist loetud noodipikkus jääb kahe antud töös kasutatava noodipikkuse vahele, saab ta endale väärtuseks suurema nendest kahest. Kui viivitus on väiksem kui 64-ndik noodi pikkusest, siis on selle pikkus 0 ehk noot algab talle eelnevaga samaaegselt. Kui failis esines taktimõõdu muutusi, poolitati see nii, et jääks alles vaid kindla taktimõõduga teisendatud nootide jadad, kui selle tulemusena tekkis 30-st noodist lühemaid jadasid, siis need kustutati.



Joonis 2. Noodikõrgused taktivektorite (oranž) ja nootide (must) kaupa. Tumedam värv näitab suuremat väärtust (0-1).

Taktivektorite leidmiseks jagatakse teos taktide kaupa osadeks ning iga osa kohta arvutatakse selles taktis leiduvate sama noodikõrgusega nootide pikkuste summa ja takti enda pikkuse suhe. Noodikõrguste all mõeldakse siinkohal 12 pooltooni võrra erinevat noodikõrgust, mis moodustavad oktavi.

Andmete eeltöötlus

Kõik MIDI andmed teisendati eelmises punktis kirjeldatud viisil. Lisaks transponeeriti iga teost, mida võimalik, 1 pooltooni võrra üles, et treeningandmeid oleks rohkem ning et närvivõrk õpiks ära tundma noodikõrguste suhtelisi vahesid ehk intervalle. Seejärel teisendatakse kõik noodikõrgused, noodi pikkused ja viivitused *one-hot* vektoriteks ning valjus pidevaks suuruseks 0-1.

Andmete jaotus

Et treenida mudeleid genereerima muusikat mitme erineva helilooja ja stiili sarnaselt ning neid omavahel võrrelda, on andmestikeks valitud järgnevad:

- Chopini teosed
- Beethoveni teosed
- Schuberti teosed
- Bachi teosed
- Romantism (Lizst, Mendelsohn, Tšaikovski, Chopin)
- Klassitsism (Clementi, Haydn, Mozart)
- Kõik koos (v.a Bach)

Andmestik	Treeninghulga suurus nootides	Valideerimishulga suurus nootides
Chopin	74075	11280
Beethoven	99602	11603
Schubert	80348	11421
Bach	190131	25689
Romantism	141550	22622
Klassitsism	103006	15368
Kõik (v.a Bach)	486580	74814

Tabel 1. Treening- ja valideerimishulkade suurus.

Andmestikud on jaotatud treening- ja valideerimisandmeteks, nii et valideerimisandmete osakaal oleks ligikaudu 15% (tabel 1), täpne jaotus pole võimalik kuna treenimiseks on vaja järjestikuseid näiteid (täpsemalt treenimise peatükis).

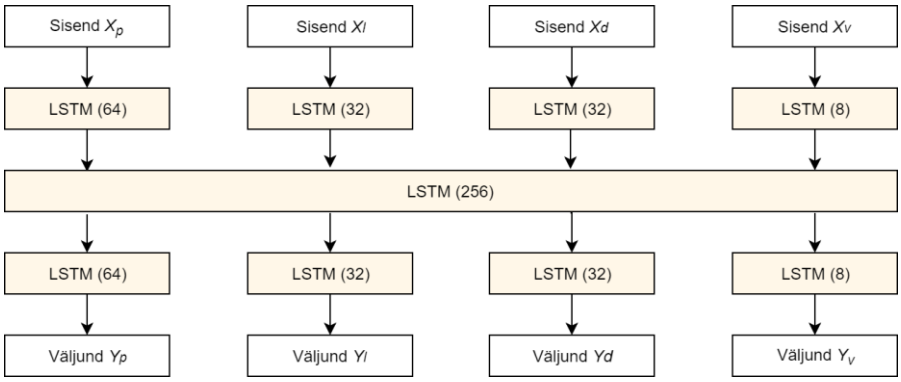
2.2 Rekurrentsetel närvivõrkudel põhinevad mudelid

Siin peatükis kirjeldatakse ülesande jaoks loodud kolme mudelit, nende treenimiskäike ja tulemusi.

Noodipõhine mudel

Noodipõhine mudel on antud ülesande jaoks loodud LSTM kihtidel põhinev tehisnärvivõrk, mille eesmärgiks on etteantud nootide jada X põhjal ennustada järgmist nooti Y.

Siin käsitlevate muusikaliste suurustega määratud teose saab kirja panna nelja järjendina: noodikõrgused (p), noodipikkused (l), viivitused (d), valjused (v). Selliselt on teose i-s noot määratud suurustega p_i, l_i, d_i ja v_i ja mudeli eesmärgiks on sisendite $X_p = (p_1, p_2, .., p_{i-1})$, $X_l = (l_1, l_2, .., l_{i-1})$, $X_d = (d_1, d_2, .., d_{i-1})$, $X_v = (v_1, v_2, .., v_{i-1})$ põhjal luua väljundid $Y_p = p_i$, $Y_l = d_i$, $Y_d = d_i$ ja $Y_v = v_i$. Kuna iga väljund on sõltuv kõigist sisenditest, siis loodi LSTM kihtidest struktuur, kus kõik sisendid ühendatakse ühte suurde LSTM kihti (joonis 3).



Joonis 3. Noodipõhise mudeli lihtsustatud struktuur. Sulgudes on märgitud LSTM ühikute arv kihis.

Väljundid Y_p, Y_l ja Y_d on mitmeklassilised klassifitseerimised ja on ühendatud neile eelnevate LSTM kihtidega tihedalt läbi *dropout* kihi. Need väljundid on tõenäosusjaotused üle oma kõigi võimalike suuruste, selleks on kasutatud aktiveerimisfunktsioonina softmax-i.

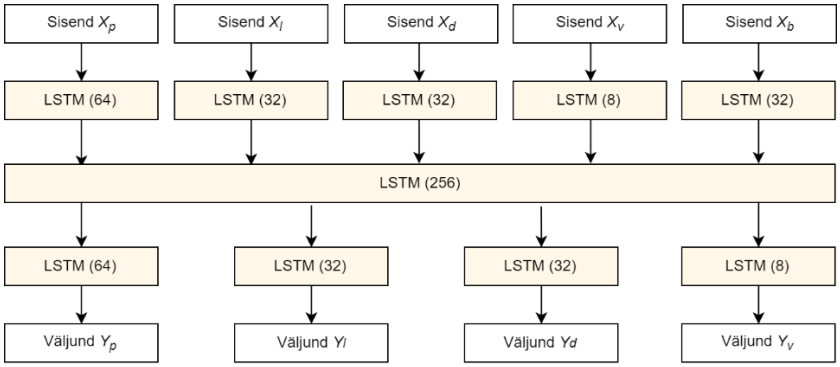
Ülesobitamise vähendamiseks lisati *dropout* kiht ka viimase ja sellele eelneva LSTM kihi vahele.

Väljund Y_v on normaliseeritud pideva suuruse ennustamine ning selleks on kasutatud sigmoid aktiveerimisfunktsiooni.

Mudeli struktuuri jaoks ja kihtide suuruste hindamiseks võeti antud ülesande jaoks eeskuju mudelitelt Performance RNN [7] ja BachProp [1].

Kombineeritud mudel

Et mudel suudaks paremini jäädvustada pikaajalist muusikalist struktuuri, lisati eelmisele mudelile veel üks sisend, milleks on takte esitavate vektorite jada (täpsemalt kirjeldatud peatükis 2) ehk sisenditele X_p, X_l, X_d ning X_v lisandus X_b (joonis 2). X_b näitab ennustatava noodi Y taktis mängivaid noodi kõrgusi ja on teiste sisenditega võrreldes ühe ajasammu võrra ees ehk kannab informatsiooni takti kohta, kuhu ennustatav noot kuulub.



Joonis 4. Kombineeritud mudeli lihtsustatud struktuur

Jooniselt 4 on näha, et kõik peale taktivektoritest sisendi lisamise on kombineeritud mudeli juures samasugune nagu noodipõhise mudeli puhul. Taktivektorid ei kanna informatsiooni ainult taktis esinevate noodikõrguste kohta vaid ka taktivahetuste kohta (kui taktivektor muutub, muutub takt).

Taktimudel

Taktimudel on mudel taktivektorite ennustamiseks. See koosneb kahest LSTM kihist ning ennustab sisendi (taktivektorite jada) põhjal järgmist taktivektorit. Taktivektori ennustamist on siinkohal vaadatud kui mitme märgendilist klassifitseerimist, kasutades selleks väljundi aktiveerimisfunktsioonina sigmoidi ja kahjufunktsioonina binaarset ristentroopiat.

2.3 Mudelite treenimine ja tulemused

Noodipõhine ja kombineeritud mudel

Noodipõhise ja kombineeritud mudeli treenimise jaoks kasutati sisendjadasid pikkusega L ning plokke (ingl *batch*) suurusega 32. Et treenida mudeleid jäädvustama ka seoseid, mille vahele jääb rohkem ajasamme kui sisendjada pikkus L, lähtestatakse LSTM ühikute olek ainult peale igat P plokki.

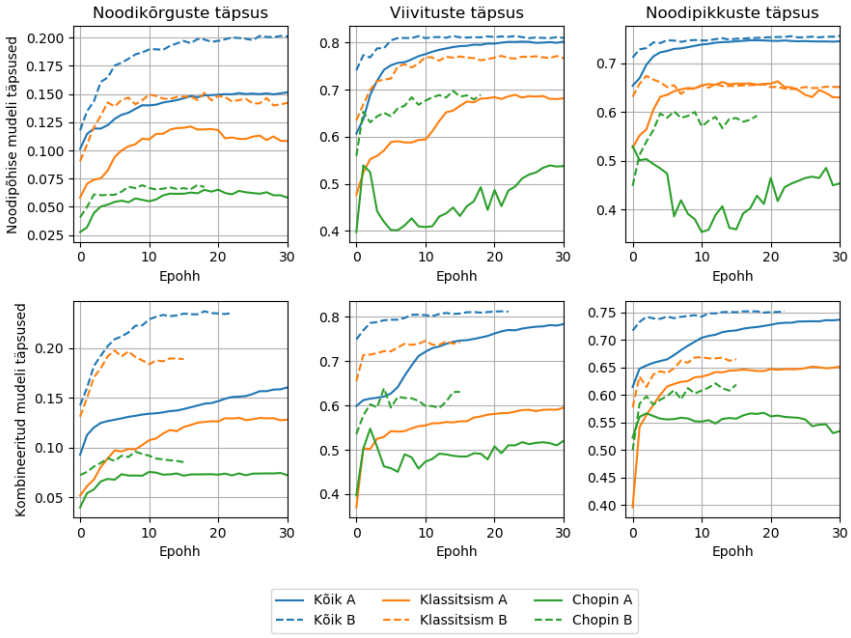
	Plokk 1		Plokk 2		…	Plokk 50	
	X	Y	X	Y		X	Y
b_1	$(p_1,..,p_{20})$	p_{21}	$(p_{21},..,p_{40})$	p_{41}		$(p_{981},..,p_{1000})$	p_{1001}
	$(l_1,..,l_{20})$	l_{21}	$(l_{21},..,l_{40})$	l_{41}		$(l_{981},..,l_{1000})$	l_{1001}
	$(d_1,..,d_{20})$	d_{21}	$(d_{21},..,d_{40})$	d_{41}		$(d_{981},..,d_{1000})$	d_{1001}
	$(v_1,..,v_{20})$	v_{21}	$(v_{21},..,v_{40})$	v_{41}		$(v_{981},..,v_{1000})$	v_{1001}

Tabel 2. Treeningandmete paigutus plokkidesse, ploki esimese elemendi b_1 näitel, kui $L = 20$ ning $P = 50$. p_i, l_i, d_i ning v_i tähistavad vastavalt i-nda elemendi (noodi) kõrgust, pikkust, viivitust ning valjust.

Selleks, et LSTM ühikute olek muutuks samamoodi nagu oleks sisse loetud üks vastava pikkusega sisendjada, on treeningandmed paigutatud plokkidesse, nii et järjestikuste plokkide samal indeksil asuvad sisendid on ühe teose järjestikused noodid (näidatud tabelis 2). Kahjufunktsioonidena kasutati klassifitseerimisväljundites ristentroopiakahju ning valjuse

väljundis keskmist ruutviga. Lisaks on lisatud väljundite Y_p, Y_l, Y_d *ja* Y_v kahjudele kordajad, vastavalt 0,7; 0,15; 0,15 ning 0,01, et arvestada klassifitseerimise kahju leidmisel klasside arvu erinevustega.

Mõlemat mudelit treeniti iga andmestikuga eraldi ning kahel korral, kasutades erinevaid L ja P väärtusi. Esimesel korral (A) kasutati suurusi $L = 20$ ja $P = 50$, teisel korral (B) $L = 30$ ja $P = 20$. B tulemused klassifitseerimises olid keskmiselt paremad igal andmestikul ning erinevate väljundite parimad klassifitseerimistulemused saavutati väiksema vahega treeningepohhidel, mis tähendab noodikõrguse klassifitseerimistäpsuse järgi kaalusid valides ka teiste väljundite paremaid täpsuseid. A ja B treeningprotsess on kujutatud joonisel 3, kust on näha, et peaaegu igal treeningepohhil on B klassifitseerimistäpsused paremad. B puhul lõpetati treenimine kui noodikõrguste klassifitseerimistäpsus 3 treeningepohhi jooksul ei paranenud. Lõplikud mudelite kaalud andmestike jaoks valiti nii, et noodikõrguste klassifitseerimistäpsus oleks suurim.



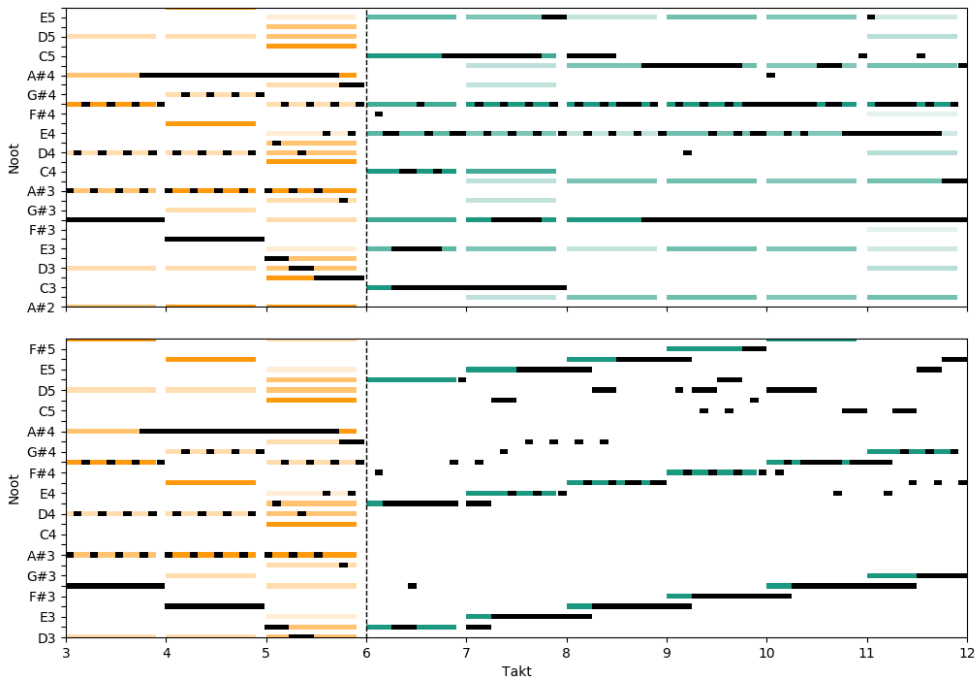
Joonis 5. A ja B treenimiskäikude võrdlus, vaadates klassifitseerimistäpsusi valideerimisandmestikel.

Kombineeritud ning noodipõhise mudeli võrdlemiseks mõõdeti mõlema viivituste, noodipikkuste ning noodikõrguste klassifitseerimistäpsust ning valjuse keskmist viga, tulemused on toodud tabelis 3. Viivituste, noodipikkuste ning valjuste juures ei ole märgatavaid erinevusi, kuid leidub suurem vahe noodikõrguste täpsuses ehk tehisnärvivõrk on õppinud kasutama taktivektoris leiduvat informatsiooni. Mõõtmisel kasutati andmeid, kus taktivektorid olid arvutatud nootide põhjal, mitte genereeritud ehk kombineeritud mudel sai informatsiooni ka ennustatava noodi kohta, kuid noodipõhine tegi ennustuse ainult eelnevate põhjal. Et hinnata kombineeritud mudeli suutlikust, kus ennustus tuleb teha ainult eelnevate nootide põhjal, tehti veel üks mõõtmine, kus ennustatava noodi taktivektor genereeriti ning saadud tulemused olid samaväärsed noodipõhise mudeli omaga.

Andmestik	Mudel	Viivituste täpsus	Noodipikkuste täpsus	Noodikõrguste täpsus	Valjuse keskmine viga
Chopin	N	0.646	0.590	0.063	0.104
	K	0.602	0.608	0.087	0.094
Schubert	N	0.800	0.743	0.158	0.078
	K	0.795	0.748	0.189	0.078
Beethoven	N	0.689	0.644	0.178	0.075
	K	0.656	0.670	0.203	0.072
Bach	N	0.815	0.654	0.161	0.207
	K	0.745	0.627	0.239	0.168
Klassitsism	N	0.761	0.657	0.151	0.071
	K	0.733	0.661	0.190	0.071
Romantism	N	0.768	0.715	0.091	0.074
	K	0.745	0.712	0.111	0.084
Kõik	N	0.814	0.755	0.203	0.064
	K	0.813	0.753	0.235	0.065

Tabel 3. Kombineeritud mudeli (K) ja noodipõhise mudeli (N) suutlikused andmestike kaupa.

Kuigi kombineeritud mudel paremaid täpsuseid ei saavutanud, võimaldab see mudel genereerida muusika vastavalt etteantud taktivektoritele (joonis 6).



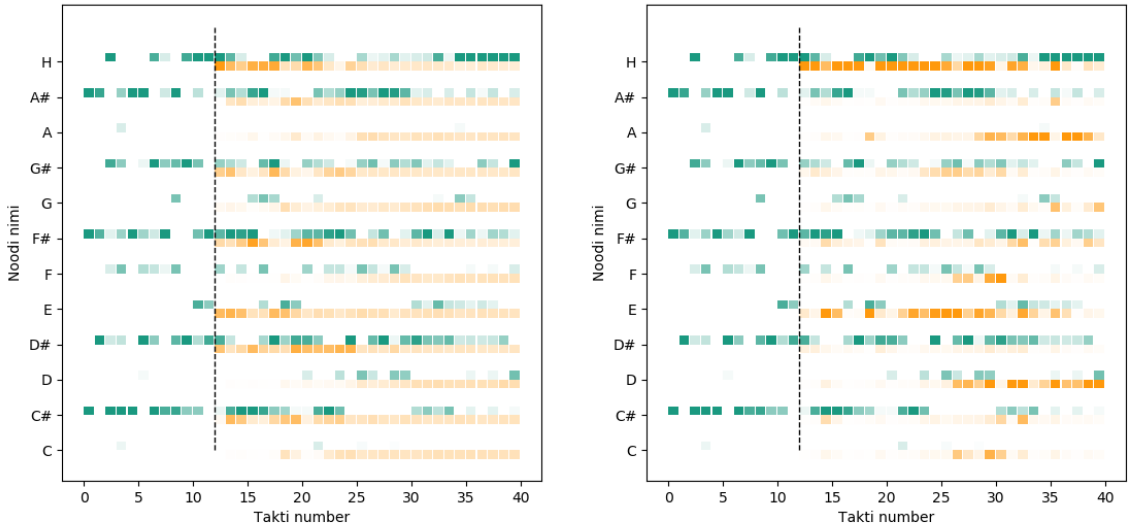
Joonis 6. Kombineeritud mudeli genereeritud nootide sõltuvus taktivektoritest. Oranžiga on tähistatud faili põhjal arvutatud taktivektorid, rohelisega on tähistatud kombineeritud

mudelile genereerimiseks ette antud taktivektorid. Esimesel juhul (ülemine) on taktivektorid genereeritud, teisel juhul (alumine) on vektorid antud käsitsi määratud väärtustega, kus i-nda vektori i-s element on 1 ja teised 0.

Jooniselt 6 on näha, et kombineeritud mudel loob muusikat vastavalt etteantud taktivektoritele, kasutades igas taktis suuremas osas noote, mis on taktivektoritena ette antud.

Taktivektorite mudel

Taktivektorite genereerimiseks loodud mudeli treenimiseks kasutati 16 vektori pikkuseid sisendjadasid ning LSTM ühikute olek lähtestati peale iga sellist jada. Iga andmestiku peal treeniti mudelile eraldi kaalud, ning kaalud salvestati failidesse.



Joonis 7. Genereeritud taktivektorite (oranž) võrdlus tegelikega (roheline) enne (vasakul) ja pärast (paremal) skaleerimist, joonega on tähistatud genereerimise algus. Tumedam värv näitab suuremat väärtust (0-1). Põhineb failil *schuim-3.mid* ja genereeritud Schuberti teoste peal treenitud taktivektorite mudeliga.

Joonise 7 vasakpoolsel osal on näha, et kuigi alguses ennustab mudel taktivektorite väärtuseid üsna täpselt, muutub väljund peale mõningaid ennustusi ebatäpseks ja taktivektori kõik suurused esinevad ühtlaselt. See võib osaliselt olla põhjustatud sellest, et tulenevalt MIDI failide nootide teisendamisega tekkivatest ebatäpsustest ei kattu taktivektorite jaoks arvestatud taktid tegelikega, mis tõttu esineb neis rohkem noote kui peaks ning suuremalt põhjustatud sellest, et iga järgnev ennustus põhineb kasvaval hulgal eelnevatel ennustustel, mille täpsus selletõttu aina kahaneb. Probleemi vähendamiseks skaleeriti iga ennustatud väärtust valemis 4 näidatud viisil ning seejärel tehti iga ennustatud taktivektori 5 kõige väiksemat väärtust nulliks.

$$t_i = \frac{e^{t_i * 5}}{\max(e^{t_1 * 5}, e^{t_{1+1} * 5}, \ldots, e^{t_j * 5})} \quad (2)$$

kus t_i on i-s element ning j on taktivektori pikkus ehk antud juhul 12.

Andmestik	Chopin	Schubert	Beethoven	Klassitsism	Romantism	Kõik
Täpsus	0.425	0.481	0.519	0.447	0.419	0.465

Tabel 4. Täpsused andmestike kaupa valideerimishulgal.

Taktivektorite mudeli täpsuse hindamiseks võrreldi ennustatud ja tegeliku vektori kolme kõige suurema väärtuse indeksitest koostatud hulkasid. Ennustus loeti täpseks kui need hulgad olid võrdsed. Täpsused on toodud tabelis 4.

2.4 Muusika genereerimine

Siin peatükis on selgitatud, kuidas muusikat mudelite abil genereeritakse ning millised probleemi mudelite poolt loodud muusikas esinevad.

Noodipõhine mudel

Noodipõhise mudeliga muusika genereerimiseks antakse mudelile ette $n > 0$ pikkused sisendjadad X_p, X_l, X_d ning X_v, mille põhjal mudel ennustab järgmise noodi Y_p, Y_l, Y_d *ja* Y_v. Sisendjadadele lisatakse vastavad ennustatud väärtused ning saadud jadade põhjal ennustab mudel järgmise noodi. Protsess kordub kuni soovitud pikkusega nootide jada on saavutatud.

Nootide ennustamisel muusika genereerimise jaoks lähtestatakse LSTM ühikute olek peale igat ennustust ning sisendjada pikkus on määratav kasutaja poolt, pikema jada korral jäetakse alles määratud arv viimast. Sisendjada pikkus määrab, kui paljude eelnevate nootide põhjal tehakse ennustus, sellest tulenevalt mõjutab see oluliselt genereeritud väljundit.

Lisaks kasutatakse klassifitseerimisväljundite klassi määramise juures temperatuuri T (käesolevas töös kasutusel valemis 3 näidatud viisil), mis muudab võimalike klasside tõenäosusjaotuse ühtlasemaks (kui $T > 1$) või vähemühtlasemaks ($T < 1$) [15].

$$p(z_i) = \frac{e^{\log(z_i)/T}}{\Sigma_{j=1}^{K} e^{\log(z_j)/T}}, \text{ kui } T \neq 1 \quad (3)$$

kus K on klasside arv ja $p(z_i)$ on z_i tõenäosus.

Klass valitakse juhuslikult saadud tõenäosusjaotuse järgi. Kui $T = 1$, valitakse kõige suurem tõenäosusega klass.

Kombineeritud mudel

Kombineeritud mudeliga muusika genereerimiseks luuakse kõigepealt taktivektorid, taktivektorid saab luua olemasoleva MIDI faili põhjal või genereerida kasutades taktivektorite mudelit. Viimase kasutamisel leitakse kõigepealt kombineeritud mudelile sisendiks antud nootide põhjal taktivektorid, seejärel antakse saadud taktivektorid sisendiks taktivektorite mudelile, kust saadud väljundi lisamisel sisendisse protsessi kordamisel, saadakse soovitud arv taktivektoreid.

Nootide genereerimine käib analoogselt noodipõhise mudeliga, aga on lisatud sisendina taktivektorid X_b. Esijalgsed taktivektorid on määratud sisendile antud nootide jadaga. Uus taktivektor antakse sisendile alles siis kui takt vahetub ehk taktivektorite sisendile lisatakse taktivektorite jada X_b viimane element, välja arvatud siis, kui takt on vahetunud, sellisel

juhul lisatakse sisendile taktivektor eelnevalt loodud järjendist. Taktide vahetumise jälgimiseks kasutatakse genereeritud nootide viivitusi: kui viivituste summa on suurem kui takti pikkus, on takt vahetunud.

Genereeritud muusika

Mudelite poolt loodud muusika kohta on toodud näiteid lisas 3. Kuna kasutatud andmete esitusviis ei suutnud väljendada kõiki treeningandmetes leiduvaid noodipikkuseid, esineb genereeritud muusikas ootamatuid ning valesti kõlavaid rütmimuutusi. Lisaks esineb lugusid, kus osad jäävad pikaks ajaks korduma.

3. Programmi paigaldamine ning kasutamine

Muusika genereerimise ja loodud mudelite kasutamiseks koostati programm koos lihtsa käsurea põhise kasutajaliidesega. Siin on toodud programmi paigaldamiseks juhend ning programmi kasutamise õpetus.

3.1 Paigaldusjuhend

Programmi kasutamiseks on vaja Pythoni versiooni 3.6.3 või uuem, programmi on testitud Windows 10-ne ja Windows 8 peal.

Programmi paigaldamiseks tuleb see repositooriumist kloonida või tõmmata alla projekti repositooriumist[14]. Kloonimiseks käivitada soovitud kaustas käsureal järgmine käsk:

```
git clone https://gitlab.com/TambetU/musicgen.git
```

Programmi poolt vajalike teekide paigaldamiseks käivatada projekti kaustas järgmine käsk:

```
pip install -r requirements.txt
```

3.2 Programmi kasutamine

Programm käivitatakse käsurealt:

```
python generate_music.py
```

Programm hakkab järjest küsima genereeritava muusika jaoks sisendeid. Esiteks tuleb otsustada, kas kasutada kombineeritud või noodipõhist mudelit. Seejärel tuleb valida, millise sisendi põhjal järgmisi noote ennustada, sisendiks võivad olla järgmised:

- MIDI fail – sisendiks olevad noodid loetakse MIDI failist;
- MIDI seadmest sissemängimine – MIDI sisendist kuulatakse noote kuni määratud arv on täis, viiakse õigele kujule ja antakse mudelile sisendiks (eeldab MIDI sisendseadme ühendatust arvutiga);
- Pole sisendit – antakse mudelile programmi poolt sisendiks juhuslike nootide jada.

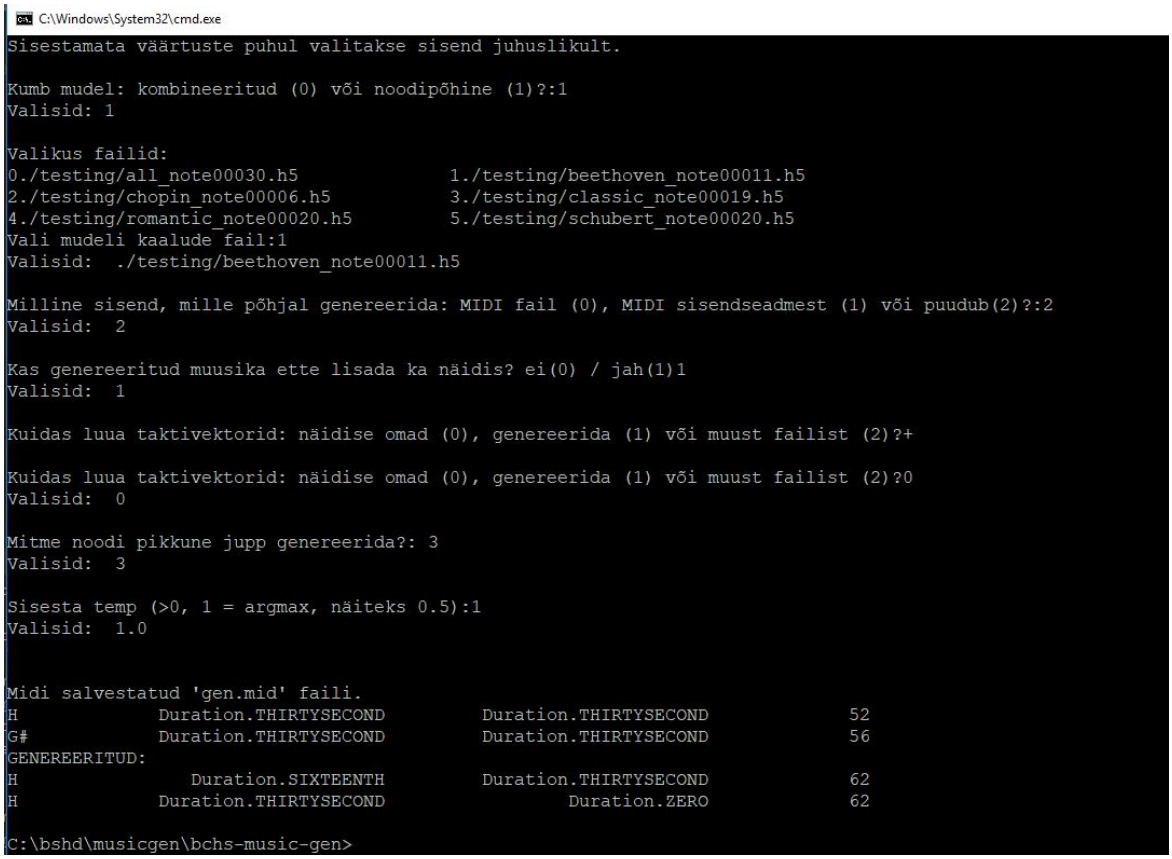
Kui valitakse kombineeritud mudel, siis tuleb lisaks määrata, kuidas luua taktivektorid. Taktivektorite loomiseks on kolm võimalust:

- arvutatakse sisendi põhjal;
- genereeritakse sisendi põhjal;
- arvutatakse mingi muu faili põhjal.

Järgmiseks tuleb määrata, kas genereeritud muusika ette väljastada ka sisendiks olnud nootide jada ning nootide ennustamiseks kasutatava nootide jada pikkus. Selle jada pikkus määrab mõjutab oluliselt genereerimise kiirust, soovitatav väärtus 50-100. Lisaks tuleb igale valitud mudelile tuleb valida kaalude fail, valitud kaalud määravad genereeritava muusika stiili. Viimaks tuleb valida genereeritava noodijada pikkus ning temperatuur. Viimase jaoks on soovitatav vahemik 0.5 – 3.

Seejärel genereeritakse määratud pikkusega nootide jada ning viiakse see MIDI kujule ning saadetakse MIDI väljundisse ning noodid väljastatakse ekraanile (joonis 8).

[14] https://gitlab.com/TambetU/musicgen



Joonis 8. Programmi kuvatõmmis.

Teine võimalus programmi kasutamiseks on muuta väärtusi projekti juurkataloogis asuvas *config.ini* failis ning määrata sealt „*cli*“ parameeter vääraks, siis algab kohe peale *generate_music.py* faili käivitamist genereerimine. Selline kasutusviis võimaldab muuta rohkem parameetreid ning programmi kiiremat kasutamist.

4. Kokkuvõte

Bakalaureusetöö eesmärk oli luua muusikat genereeriv programm, mis suudaks seda teha mitmele heliloojale ning stiilile omaselt, luues kõigepealt taktipõhise struktuuri ning selle põhjal üksikud noodid. Programmi tegemiseks oli vaja luua MIDI andmete jaoks lihtsam esitusviis, 3 rekurrentsetel tehisnärvivõrkudel põhinevat mudelit, need mudelid treenida ning luua käsurea põhine kasutajaliides.

Kõikide mudelite loomine ja treenimine õnnestus, kuid taktipõhist struktuuri kasutav mudel ei saavutanud paremaid täpsuseid nootide ennustamisel kui ainult nootide põhjal ennustusi tegev mudel. Sellegi poolest võimaldab selline mudel anda genereeritavale muusikale ette taktide kaupa noodid, mis seal esinevad.

Töö tulemusena valmis programm, mis võimaldab genereerida muusikat, kasutades kahte erinevat mudelit koos seitsme erineva stiili või helilooja teoste peal treenimise tulemusena saadud kaaludega.

Programmi edasiarendamiseks saab mudeleid treenida suurema hulga andmete peal, parandada mudelite sooritust, optimeerides hüperparameetreid ning muutes muusikaliste andmete esitusviisi selliselt, et sellega oleks võimalik katta kõik võimalikud noodipikkused ning viivitused.

5. Viidatud kirjandus

1. Colombo F, Gerstner W. BachProp: Learning to Compose Music in Multiple Styles. 2018. http://arxiv.org/abs/1802.05162 (10. jaanuar 2019)

2. Chung J, Gulcehre C, Cho K, Bengio Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, 2014. http://arxiv.org/abs/1412.3555 (24. aprill 2019)

3. Lipton ZC, Berkowitz J, Elkan C. A Critical Review of Recurrent Neural Networks for Sequence Learning, 2015. http://arxiv.org/abs/1506.00019 (8. mai 2019)

4. Hochreiter S, Schmidhuber J. LONG SHORT-TERM MEMORY. 1997. https://www.bioinf.jku.at/publications/older/2604.pdf (1 .mai 2019)

5. Graves A. Generating Sequences With Recurrent Neural Networks, 2013. http://arxiv.org/abs/1308.0850 (24. aprill 2019)

6. Eck D, Schmidhuber J. A First Look at Music Composition using LSTM Recurrent Neural Networks, 2002. http://people.idsia.ch/~juergen/blues/IDSIA-07-02.pdf (10. aprill 2019)

7. Simon I, Sageev O. Performance RNN: Generating Music with Expressive Timing and Dynamics, Magenta, 2017. https://magenta.tensorflow.org/performance-rnn (11. jaanuar 2019)

8. Bhande A. What is underfitting and overfitting in machine learning and how to deal with it. Medium. 2018. https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76 (10. mai 2019)

9. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014, 30. http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf (14. aprill 2019)

10. Mozer MC. Neural network music composition by prediction: Exploring the benefits of psychophysical constraints and multiscale processing, 1994. http://www.cs.colorado.edu/~mozer/Research/Selected%20Publications/music.html (10. jaanuar 2019)

11. Huang C-ZA, Vaswani A, Uszkoreit J, Shazeer N, Simon I, Hawthorne C, et al. Music Transformer, 2018. http://arxiv.org/abs/1809.04281 (11. jaanuar 2019)

12. Oord A van den, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, et al. WaveNet: A Generative Model for Raw Audio. 2016. http://arxiv.org/abs/1609.03499 (23. veebruar 2019)

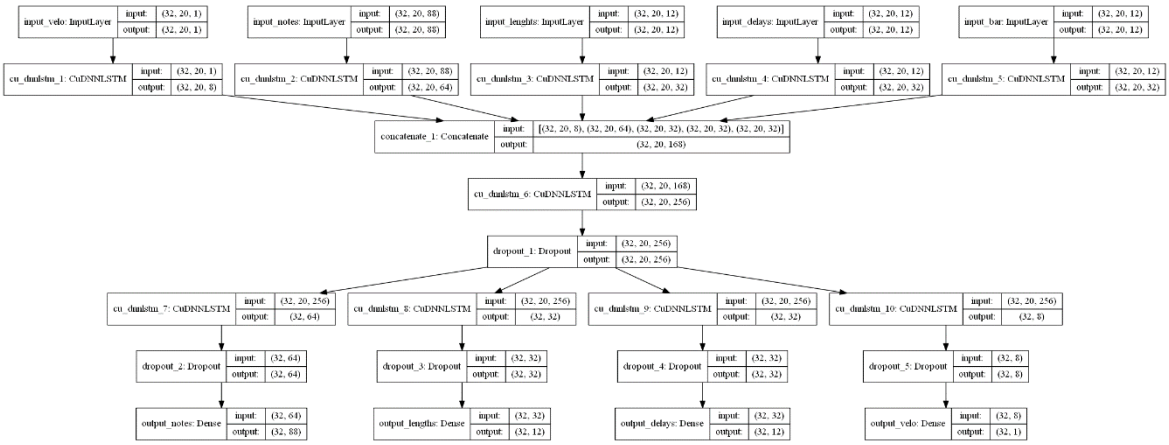
13. Chollet F, others. Keras. GitHub; 2015. https://github.com/fchollet/keras (10. jaanuar 2019)

14. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. https://www.tensorflow.org/ (10. jaanuar 2019)

15. Guo C, Pleiss G, Sun Y, Weinberger KQ. On Calibration of Modern Neural Networks, 2017. http://arxiv.org/abs/1706.04599 (9. mai 2019)

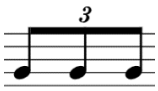
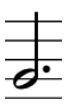
Lisad

I. Kombineeritud mudeli detailne struktuur.



Kombineeritud mudeli struktuur. Joonis loodud kasutades Kerase *plot_model* funktsiooni.

II. Kasutatud noodipikkuste ja viivituste vasted noodikirjas

Osa tervenoodist	Noodikirja vaste
0	-
1/64	
2/64	
4/64	
16/192	
8/64	
12/64	
16/64	
24/64	
32/64	
48/64	
1	

III. Programmi poolt genereeritud muusika näited

Näiteid erinevatel andmestikel treenitud mudelite loodud muusikast:

https://soundcloud.com/tambet-uutsalu/koik-kombineeritud-mudel

https://soundcloud.com/tambet-uutsalu/schubert-kombineeritud

https://soundcloud.com/tambet-uutsalu/schubert-noodipohine

https://soundcloud.com/tambet-uutsalu/chopin-noodipohine

https://soundcloud.com/tambet-uutsalu/chopin-kombineeritud

https://soundcloud.com/tambet-uutsalu/bach-kombineeritud-mudel

https://soundcloud.com/tambet-uutsalu/bach-noodipohine-mudel

https://soundcloud.com/tambet-uutsalu/romantism-kombineeritud-mudel

https://soundcloud.com/tambet-uutsalu/klassitsism-noodipohine-mudel

Ülaltoodud näited on loodud, kasutades 100 noodi pikkust sisendjada, T=0.5 ning kombineeritud mudeli puhul genereeritud taktivektoreid. Nime esimene osa näitab andmestiku, mille peal mudel treeniti. Sisendfaili nimi on kirjas teose juures.

Näide taktivektorite kasutamisest genereeritud muusika mõjutamiseks:

https://soundcloud.com/tambet-uutsalu/etteantud-taktivektoritega-klassitsism-2-sisend-genereeritu-ees

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina,

Tambet Uutsalu _____,

(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

Muusika genereerimine närvivõrkude abil
_____
_____
_____,

(lõputöö pealkiri)

mille juhendaja on

Sven Aller_____,

(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Tambet Uutsalu

10.05.2019