OULUN AMMATTIKORKEAKOULU

ÄÄNENLAADUN PARANTAMINEN NEUROVERKKOJEN AVULLA

Niklas Siltala
Opinnäytetyö
Kevät 2025
Tietotekniikan
tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tieto- ja viestintätekniikka, Insinööri

Tekijä: Niklas Siltala
Opinnäytetyön nimi: Äänenlaadun parantaminen neuroverkkojen avulla
Työn ohjaaja: Ilpo Virtanen
Työn valmistumislukukausi ja -vuosi: Kevät 2025
Sivumäärä: 39

Digitaalinen ääni, varsinkin pakattu tai peräisin vanhanaikaisista nauhoituksista, kärsii heikosta äänenlaadusta ja yksityiskohtien puutteesta. Tämän työn tavoitteena oli kehittää tekoälymalli (koneoppimismalli), joka kykeni parantamaan tällaista huonolaatuista ääntä. Erityisesti työssä pyrittiin palauttamaan äänenlaatu tilanteissa, joissa ääni oli heikentynyt digitaalisen prosessoinnin vuoksi.

Työssä hyödynnettiin syväoppimismenetelmiä, erityisesti Generatiivisia Kilpailevia Verkkoja (GAN). Mallin kehittämiseen ja opettamiseen käytettiin monipuolista äänidataa erilaisista musiikin tyylilajeista, joiden äänenlaatua heikennettiin ohjelmallisesti.

Työn tuloksena saatiin kehitettyä malli, joka kykeni havaittavasti parantamaan heikentynyttä äänenlaatua testatuilla aineistoilla. Esimerkiksi heikkolaatuisessa äänessä havaittiin selkeyden lisääntymistä ja yksityiskohtien palautumista verrattuna käsittelemättömään huonolaatuiseen ääneen. Ääniraidassa havaittiin selviä parannuksia, joita ei huonolaatuisessa äänessä ole.

Työn perusteella voidaan todeta, että syväoppiminen ja erityisesti GAN-arkkitehtuurit ovat tehokkaita menetelmiä digitaalisesti heikentyneen äänenlaadun parantamiseen. Kehitetty malli osoittautuu lupaavaksi lähestymistavaksi äänen restaurointiin. Jatkokehitysehdotuksina esitetään mallin suorituskyvyn laajempaa arviointia erityyppisillä äänimateriaaleilla sekä reaaliaikaisen suorituskyvyn parantamista.

ABSTRACT

Oulu University of Applied Sciences
Information and Communication Technology, Engineer

Author: Niklas Siltala
Title of thesis: Audio quality enhancement with neural networks
Supervisor: Ilpo Virtanen
Term and year when the thesis was submitted: Spring 2025
Number of pages: 39

Digital audio, especially when compressed or originating from old recordings, suffers from poor audio quality and a lack of detail. The goal of this work was to develop an AI model (machine learning model) capable of enhancing such low-quality audio. Specifically, the work aimed to restore audio quality in situations where the audio had been degraded due to digital processing.

The work utilized deep learning methods, particularly Generative Adversarial Networks (GANs). Diverse audio data from various music genres was used to develop and train the model, with the audio quality being programmatically degraded.

As a result of the work, a model was developed that was noticeably capable of improving degraded audio quality on the tested datasets. For instance, in low-quality audio, an increase in clarity and the restoration of details were observed compared to the unprocessed low-quality audio. Clear improvements were observed in the audio track that are not present in the low-quality audio.

Based on the work, it can be concluded that deep learning, and particularly GAN architectures, are effective methods for improving digitally degraded audio quality. The developed model proves to be a promising approach for audio restoration. Suggestions for further development include a broader evaluation of the model's performance with different types of audio material and improving real-time performance.

SISÄLLYS

1 JOHDANTO

1.1 Aiheen esittely ja taustaa

Internetissä on valtava määrä ääniklippejä ja videoita, mutta valitettavasti niiden äänenlaatu ei aina ole paras mahdollinen. Monissa tapauksissa taustamelu, häiriöt ja matala bittinopeus voivat heikentää kuuntelukokemusta merkittävästi. Tämä on erityisen ongelmallista, kun kyseessä on tallenteita monesta erilaisesta mediasta, joissa äänen selkeys on ensiarvoisen tärkeää.

Neuroverkot, erityisesti generatiiviset adversaariset verkot (GAN), ovat tuoneet merkittäviä edistysaskeleita kuvanlaadun parantamisessa. GAN-menetelmä perustuu kahden neuroverkon yhteistyöhön: generatiivinen verkko tuottaa parannettua sisältöä syötteen perusteella, kun taas diskriminatiivinen verkko arvioi niiden laatua ja pyrkii erottamaan ne todellisesta datasta. Tämä vuorovaikutus johtaa jatkuvaan parannukseen, kun generatiivinen verkko oppii tuottamaan yhä realistisempaa ja laadukkaampaa ääntä. (Goodfellow ym. 2014)

Vaikka GAN-tekniikkaa on käytetty laajalti kuvien laadun parantamiseen, sen soveltaminen äänen parantamiseen on myös lupaavaa. Esimerkiksi GAN-menetelmällä voidaan vähentää taustamelua ja korjata äänen vääristymiä, mikä tekee äänestä selkeämpää ja miellyttävämpää kuunnella. Tämä menetelmä voi olla erityisen hyödyllinen vanhojen äänitallenteiden restauroinnissa tai live-lähetysten laadun parantamisessa. (TecnoDigital 25.2.2024)

Neuroverkot tarjoavat tehokkaita työkaluja äänenlaadun parantamiseen. GAN-menetelmän avulla voidaan saavuttaa merkittäviä parannuksia, jotka tekevät kuuntelukokemuksesta miellyttävämpää ja informatiivisempaa. Tulevaisuudessa näiden teknologioiden kehitys voi johtaa entistä parempiin ratkaisuihin ja laajempiin sovelluksiin eri aloilla.

1.2 Tavoite

Tämän projektin tavoite on kehittää neuroverkko, joka kykenee parantamaan sekunnin pituisia ääniklippejä alkuperäisen tasolle hyödyntäen generatiivisia kilpailevia verkkoja (GAN). Tämä on erittäin haastava tehtävä, sillä lyhyet ääniklipit sisältävät usein paljon melua ja häiriöitä, jotka voivat heikentää niiden laatua merkittävästi.

1.3 Rajaus

Projektin resurssivaatimukset, erityisesti laskentateho ja tallennustila, ovat merkittävästi rajoittaneet projektin laajuutta. Äänidatan pituuden rajoittaminen yhteen sekuntiin mahdollistaa nopeamman koulutusprosessin, mikä on kriittistä käytettävissä olevien resurssien puitteissa samalla kun se nopeuttaa tulosten saantia. Tulevaisuudessa, kun resursseja ja aikaa on enemmän saatavilla, voimme laajentaa projektia ja käsitellä pidempiä ääninäytteitä.

1.4 Rakenne

Projektissa käytettävä aineisto koostuu hyvistä ja huonoista ääninäytteistä. Projekti sisältää myös generaattorin ja diskriminaattorin (mallit). Ääninäytteet ovat sekunnin pituisia, ja nämä näytteet kerätään erilaisista lähteistä. Ne ovat korkealaatuisia, mutta ne pitää esikäsitellä.

Koulutusdatan esikäsittelyyn kuuluu:

- Ääninäytteen näytteenottotaajuuden muuttaminen (alasnäytteistys).
- Tulevaisuudessa myös kohinaa lisätään ja Plogue™ Chipcrusher DAC (digital analog converter) emulaattoria käytetään tuomaan realistista kohinaa.

Tämän jälkeen näytteet syötetään neuroverkolle koulutusdataksi. Koulutuksen aikana diskriminaattori koulutetaan ensin näytteillä ja arvioidaan sen kyky tunnistaa huonolaatuisia ääninäytteitä. Sen jälkeen huonolaatuiset ääninäytteet syötetään generaattorille, jotka yrittävät syötteiden pohjalta luoda laadukkaalta ääninäytteeltä muistuttavia tulosteita muuttaen näytteiden äänenvoimakkuustasoja, hämäten diskriminaattoria. Näin molemmat täydentävät toisiaan ja tulostukset generaattorilta paranevat jokaisella sukupolvella.

2 KIRJALLISUUSKATSAUS

2.1 Äänenlaadun parantaminen yleisesti

Jotta äänenlaatua voidaan parantaa on ensin otettava selville mitä on digitaalinen ääni. Digitaalinen ääni koostuu seuraavista elementeistä:

Näytteenottotaajuudesta (englanniksi sample rate), joka määrittää, kuinka monta kertaa sekunnissa ääni mitataan ja tallennetaan digitaaliseen muotoon. Se ilmaistaan hertseinä (Hz). Esimerkiksi CD-laatuisessa äänessä näytteenottotaajuus on 44,1 kHz, mikä tarkoittaa, että ääni mitataan 44 100 kertaa sekunnissa. (Federal Agencies Digitization Guidelines Initiative 2025)

Näytteiden tarkkuudesta (englanniksi bit depth), joka määrittää, kuinka monta bittiä käytetään kunkin näytteen tallentamiseen. Se vaikuttaa äänen dynaamiseen alueeseen ja tarkkuuteen. Esimerkiksi 16-bittinen ääni voi esittää 65 536 erilaista äänenvoimakkuustasoa, kun taas 24-bittinen ääni voi esittää yli 16 miljoonaa tasoa. (Toft, R. 2024)

Kanavista (englanniksi channels), jotka tuovat kuulijalle paremman kuuntelukokemuksen. Monofoninen (yksi kanava), stereofoninen (kaksi kanavaa) tai monikanavainen (useita kanavia). Stereofonisessa äänessä on vasen ja oikea kanava, ja ne eriävät toisistaan. Monikanavaisessa äänessä, kuten surround-äänessä, on viisi tai enemmän kanavia, jotka ympäröivät kuuntelijan. (Wildlife Acoustics 2025)

Näytteistä (englanniksi sample), jotka ovat yksittäisiä mittauksia, jotka tallennetaan näytteenottotaajuuden ja näytteen tarkkuuden mukaisesti. Näytteet muodostavat digitaalisen äänisignaalin, joka voidaan toistaa ja käsitellä. (Toft, R. 2024)

Nämä kaikki vaikuttavat äänenlaatuun. Tässä projektissa keskitytään nostamaan näytteenottotaajuutta 11 025 Hz:stä 44 100 Hz:iin. Tavoitteena on, että ääni kuulostaa selkeämmältä ja yksityiskohtaisemmalta.

2.2 Neuroverkot äänenkäsittelyssä

Neuroverkkojen soveltaminen äänenkäsittelyyn on tuonut merkittäviä edistysaskeleita. Näiden verkkojen avulla voidaan käsitellä ja analysoida monimutkaisia ääniä tehokkaasti. Esimerkiksi niitä voidaan käyttää moniin eri tarkoituksiin, kuten taustamelun poistamiseen, äänen parantamiseen ja puheentunnistukseen. Yksi merkittävä esimerkki neuroverkkojen soveltamisesta äänenkäsittelyyn on RNNoise. (Xiph.Org 2017)

Ennen kuin neuroverkko voi käsitellä ääntä, raaka äänisignaali on muunnettava muotoon, joka soveltuu paremmin verkon syötteeksi. Yleisempi lähestymistapa on muuntaa ääni spektrogrammiksi tai joksikin sen johdannaiseksi (kuten mel-spektrogrammiksi). (Abdul & Al-Talabani 2022)

Spektrogrammi on visuaalinen esitys äänen taajuussisällön muutoksesta ajan funktiona. Se lasketaan tyypillisesti jakamalla äänisignaali lyhyisiin, päällekkäisiin aikakehyksiin ja laskemalla kullekin kehykselle Fourier-muunnos (usein lyhytaikainen Fourier-muunnos, STFT). Tämä muunnos paljastaa kussakin aikaikkunassa esiintyvät taajuuskomponentit ja niiden voimakkuudet. Mel-spektrogrammi on jatkojalostettu versio, jossa taajuusakseli on skaalattu ihmisen kuulon epälineaarisen herkkyyden mukaisesti (mel-asteikko). Nämä spektrogrammit muistuttavat kuvadataa, mikä mahdollistaa kuvankäsittelystä tuttujen neuroverkkoarkkitehtuurien, kuten konvoluutioneuroverkkojen, hyödyntämisen. (Durak & Arikan 2003)

2.3 Muita aiempia tutkimuksia ja niiden tulokset

Jyväskylän yliopistossa tehty tutkimus käsitteli keinotekoisten neuroverkkojen hyödyntämistä lintujen automaattisessa tunnistamisessa äänen perusteella. Tutkimuksessa havaittiin, että neuroverkot voivat tunnistaa lintulajeja tehokkaammin kuin perinteiset menetelmät, mikä voi olla hyödyllistä luonnonsuojelussa ja lintuharrastuksessa. (Sintonen, L 2018)

Phonak on käyttänyt koneoppimista kuulokojeiden AutoSense OS -käyttöjärjestelmän kouluttamiseen. Tämä järjestelmä luokittelee ääniympäristöjä ja säätää kuulokojeen parametreja automaattisesti, parantaen käyttäjän kuuntelukokemusta. (Phonak 2025)

3 MENETELMÄT

3.1 Käytetyt neuroverkkotyypit ja niiden perustelut

Neuroverkot ovat laskennallisia malleja, jotka jäljittelevät ihmisaivojen rakennetta ja toimintaa. Ne koostuvat toisiinsa kytketyistä "neuroneista", jotka ovat järjestettyjä kerroksiin, ja ne oppivat tunnistamaan monimutkaisia kuvioita ja yhteyksiä suuren tietomäärän perusteella erilaisten tehtävien suorittamiseksi, kuten luokitteluun tai ennustamiseen. Konvolutiiviset neuroverkot (englanniksi convolutional neural networks) ovat erityinen neuroverkkojen tyyppi, jotka ovat osoittautuneet erittäin tehokkaiksi visuaalisen tiedon, kuten kuvien ja videoiden, käsittelyssä. Niiden rakenne sisältää erikoiskerroksia, jotka pystyvät automaattisesti oppimaan ja tunnistamaan visuaalisia piirteitä kuvista, kuten reunoja, kulmia ja monimutkaisempia muotoja, mikä tekee niistä standardiratkaisun esimerkiksi kuvantunnistuksen, objektien havaitsemisen ja kuva-analyysin tehtävissä. Nämä piirteet esiintyvät myös äänisignaaleissa ja projektissa pyrin hyödyntämään niitä. (MIT News 2017)

Käytän generatiivisia kilpailevia verkkoja (Generative Adversarial Networks, GAN), joissa kaksi neuroverkkoa – generaattori (englanniksi generator) ja diskriminaattori (englanniksi discriminator) – kilpailevat keskenään. Äänenparannuksen kontekstissa generaattorin tavoitteena on muuntaa heikkolaatuinen tai kohinainen äänisignaali korkealaatuiseksi ääneksi, kun taas diskriminaattori pyrkii erottamaan aidot korkealaatuiset ääninäytteet generaattorin tuottamista näytteistä. Projektissa olevat mallit hyödyntävät useita edistyneitä neuroverkkokomponentteja tämän tehtävän suorittamiseksi tehokkaasti. Tarkastellaan näiden komponenttien tyyppejä ja syitä niiden käyttöön.

Generaattori

Generaattorin päätavoite on oppia muuttamaan syötetty ääni uskottavaksi ja laadukkaaksi tavoiteääneksi. Se käsittelee ääntä erityisten suodattimien eli konvoluutiokerrosten avulla. Nämä suodattimet tarkastelevat ääntä pieninä palasina, joiden kokoa säädetään ytimen koon avulla. Jotta malli pystyisi ymmärtämään äänen pidemmän aikavälin rakenteita tehokkaasti, se hyödyntää dilataatiota, joka mahdollistaa laajempien alueiden tarkastelun harvemmilla askelilla. Käsittelyn aikana äänen pituuden hallinta varmistetaan täytön avulla, eli äänisignaaliin lisätään näytteitä jotta signaalin pituus on vakio.

Generaattorin vakauden parantamiseksi ja ei-toivottujen tyylivaihteluiden poistamiseksi käytetään instanssinormalisointia, joka tasapäistää piirteet kunkin yksittäisen ääninäytteen sisällä. Konvoluutiokerrosten jälkeen aktivaatiofunktio, kuten PReLU, päättää, mitkä löydetyt piirteet ovat merkityksellisiä jatkossa. Lisäksi generaattoriin kuuluu huomiomekanismi, joka oppii dynaamisesti painottamaan eri piirrekanavia niiden tärkeyden perusteella, auttaen mallia keskittymään olennaisimpaan äänen generoinnin aikana.

Generaattorin ytimen muodostavat Residual In Residual Block (RIRB) -lohkot. Nämä lohkot sisältävät useita perussuodatinrakenteita ja huomio-osiota. Niiden kriittisin osa on residuaalinen yhteys, jossa lohkon alkuperäinen syöte lisätään suoraan sen lopputulokseen. Nämä yhteydet ovat tärkeitä syvempien verkkojen tehokkaan oppimisen kannalta, sillä ne auttavat oppimissignaalin kulkua ja mahdollistavat monimutkaisempien äänenmuutosten oppimisen.

Generaattorin erittäin tärkeä piirre on sen globaali residuaalinen yhteys, joka ulottuu koko verkkoon. Tämä tarkoittaa, että generaattori ei pyri tuottamaan koko tavoiteääntä alusta loppuun, vaan sen sijaan oppii ainoastaan sen erotuksen tai korjauksen, joka tarvitaan alkuperäisen äänen muuttamiseksi tavoiteääneksi. Generaattori siis keskittyy oppimaan, mitä alkuperäiseen signaaliin pitää lisätä tai siitä poistaa. Tämä "erotuksen oppimisen" strategia on

usein merkittävästi vakaampi ja tehokkaampi erityisesti äänenlaadun parantamiseen tähtäävissä tehtävissä.

Diskriminaattori

Diskriminaattori on eräänlainen "tuomari" tai "tarkastaja" GAN-pohjaisessa tekoälyjärjestelmässä. Generaattori yrittää luoda aidolta kuulostavaa ääntä ja Diskriminaattorin päätehtävä on oppia erottamaan Generaattorin luomat väärennökset aidoista, todellisista ääninäytteistä. Nämä kaksi tekoälymallia koulutetaan yhdessä: Generaattori yrittää tehdä yhä laadukkaampia ääninäytteitä ja Diskriminaattori yrittää tulla yhä paremmaksi paljastamaan oikeat väärästä.

Diskriminaattori käsittelee ääntä käyttämällä toistuvia rakennuspalikoita. Nämä palikat koostuvat yksiulotteisista konvoluutiokerroksista, vakauttajista kuten spektrinormalisoinnista ja päätöksenteko-osista kuten LeakyReLU. Palikat analysoivat ääntä etsimällä siitä tiettyjä kuvioita ja piirteitä, vähän kuin tutkisivat ääniraitaa pienissä osissa kerrallaan. Se tekee tämän tehokkaasti hyppimällä äänessä eteenpäin, jotta tiedosta tulee tiiviimpää, ja samalla "venyttämällä katsettaan" tunnistaakseen kuvioita eri mittakaavoissa eli sekä lyhyillä että pidemmillä äänipätkillä. (Papers With Code 2025)

Tällaisten järjestelmien kouluttaminen voi olla epävakaata, mutta tämä Diskriminaattori sisältää erityisiä tekniikoita vakauden parantamiseksi. Se käyttää menetelmiä, kuten spektrinormalisointia, jotka toimivat laskennan "vakauttajina" ja estävät lukuja kasvamasta liian suuriksi, mikä tekee harjoittelusta tasaisempaa. Lisäksi se hyödyntää tietynlaisia tapoja käsitellä tietoa matkan varrella, jotta prosessi pysyy hallinnassa.

Ennen lopullisen päätöksen tekemistä Diskriminaattorilla on kyky keskittyä niihin äänestä löytyviin yksityiskohtiin, jotka parhaiten paljastavat onko ääni aito vai keinotekoinen (tämä on sen "tarkkaavaisuusmekanismi"). Lopuksi se tiivistää kaiken analysoimansa tiedon koko äänipätkästä yhteenvedoksi ja antaa sen

perusteella lopullisen arvionsa siitä, kuinka todennäköisesti syötetty ääni on todellinen.

3.2 Äänidatan esikäsittely

Alkuperäisistä äänitiedostoista luodaan pareja: Alkuperäinen korkealaatuinen ja prosessoitu heikennetty matalalaatuinen ääni. Keskeisiä esikäsittelyvaiheita ovat äänitiedostojen muuttaminen yksikanavaiseksi, näytteenottotaajuuden alentaminen ja äänitiedoston pituuden vakiointi.

Äänitiedostot haetaan hakemistosta ja ne ladataan torchaudio.load-funktion avulla. Tämä palauttaa äänidatan tensorimuodossa ja alkuperäisen näytteenottotaajuuden.

Jotta saataisiin luotua matalalaatuinen versio alkuperäisestä korkealaatuisesta äänestä (jota käytetään mallin syötteenä), korkealaatuinen ääni käsitellään alentamalla sen näytteenottotaajuutta. Tätä prosessia nimitetään uudelleennäytteistykseksi.

Sen jälkeen, kun äänitensoreiden laatu on heikennetty, ne muunnetaan monikanavaisesta muodosta yksikanavaiseksi monoääneksi. Tämä vaihe on tarpeen mallin tehokkaan toiminnan vuoksi ja syötteen yksinkertaistamiseksi, koska mallin arkkitehtuuri on suunniteltu käsittelemään yhtä kanavaa. Äänen muuntaminen monoääneksi yksinkertaistaa syötettä merkittävästi ja auttaa mallia keskittymään äänen sisältöön sen sijaan, että sen tarvitsisi käsitellä useampaa samanaikaista kanavaa.

On tärkeää, että kaikki äänitensorit ovat samanpituisia. Äänidatan maksimipituudeksi asetetaan 44100 näytettä. Lyhyemmät äänitiedostot täytetään hiljaisuudella loppuun asti, kunnes ne saavuttavat tämän pituuden. Pidemmistä äänitiedostoista puolestaan otetaan vain alkuosa maksimipituuden verran. Näillä toimenpiteillä varmistetaan yhtenäinen pituus kaikille mallille annettaville äänisignaaleille.

Lopuksi sekä korkealaatuinen että matalalaatuinen äänitensori siirretään määritellylle laitteelle, kuten GPU:lle, laskennan nopeuttamiseksi.

Metodi palauttaa parin (englanniksi tuple), joka sisältää kaksi muuta paria: ensin korkealaatuisen äänitensorin ja sen alkuperäisen näytteenottotaajuuden, ja toisena matalalaatuisen äänitensorin ja sen luonnissa käytetyn alhaisemman näytteenottotaajuuden.

```
from torch.utils.data import Dataset
import torch.nn.functional as F
import torch
import torchaudio
import os
import random
import torchaudio.transforms as T
import AudioUtils

class AudioDataset(Dataset):
    audio_sample_rates = [11025]
    MAX_LENGTH = 44100  # Define your desired maximum length here

    def __init__(self, input_dir, device):
        self.input_files = [os.path.join(root, f) for root, _, files in os.walk(input_dir) for f in
files if f.endswith('.wav')]
        self.device = device

    def __len__(self):
        return len(self.input_files)

    def __getitem__(self, idx):
        # Load high-quality audio
        high_quality_audio, original_sample_rate = torchaudio.load(self.input_files[idx], normal-
ize=True)

        # Generate low-quality audio with random downsampling
        mangled_sample_rate = random.choice(self.audio_sample_rates)
        resample_transform_low = torchaudio.transforms.Resample(original_sample_rate, mangled_sam-
ple_rate)
        low_quality_audio = resample_transform_low(high_quality_audio)

        resample_transform_high = torchaudio.transforms.Resample(mangled_sample_rate, original_sam-
ple_rate)
        low_quality_audio = resample_transform_high(low_quality_audio)

        high_quality_audio = AudioUtils.stereo_tensor_to_mono(high_quality_audio)
        low_quality_audio = AudioUtils.stereo_tensor_to_mono(low_quality_audio)


        if high_quality_audio.shape[1] < self.MAX_LENGTH:
            padding = self.MAX_LENGTH - high_quality_audio.shape[1]
            high_quality_audio = F.pad(high_quality_audio, (0, padding))
        elif high_quality_audio.shape[1] > self.MAX_LENGTH:
            high_quality_audio = high_quality_audio[:, :self.MAX_LENGTH]

        if low_quality_audio.shape[1] < self.MAX_LENGTH:
            padding = self.MAX_LENGTH - low_quality_audio.shape[1]
```

```
            low_quality_audio = F.pad(low_quality_audio, (0, padding))
        elif low_quality_audio.shape[1] > self.MAX_LENGTH:
            low_quality_audio = low_quality_audio[:, :self.MAX_LENGTH]

        high_quality_audio = high_quality_audio.to(self.device)
        low_quality_audio = low_quality_audio.to(self.device)

        return (high_quality_audio, original_sample_rate), (low_quality_audio, mangled_sample_rate)
```

KUVA 1. Koodi projektin datan esikäsittelystä

3.3 Neuroverkon koulutusprosessi

Koulutusprosessi alkaa jakamalla koulutusdata sopiviin eriin, jotta saadaan muistin käyttö optimoitua. Tämä data annetaan diskriminaattorille ja generaattorille koulutusiteraation (englanniksi training iteration tai epoch) aikana.

Mallin oppimiseen vaikuttavat häviöfunktiot. Koulutusskripti käyttää binääristä ristiinentropiahäviötä (englanniksi BCE with logits loss) sekä generaattorin että diskriminaattorin häviöfunktiona. Tämä on yleinen valinta GAN-verkkojen koulutuksessa, sillä se mahdollistaa diskriminaattorin antaman binäärisen luokittelun (aito/väärennös) vertaamisen todelliseen.

Koulutuksessa käytetään Adam-nimistä optimoijaa, joka on yleinen ja hyvä tapa saada neuroverkko oppimaan paremmin. Oppimiseen vaikuttaa myös oppimisnopeus. Tätä säädetään automaattisesti pienemmäksi, jos malli ei enää kehity merkittävästi. Näin malli oppii paremmin, välttää ylikoulutuksen ja sen sisäiset osat (painot) asettuvat oikein. (Kingma & Ba 2017)

Varsinainen koulutus tapahtuu iteratiivisessa silmukassa. Jokaisella iteraatiolla (eli epokilla) käydään läpi koko koulutusdata. Yhden epokin aikana data jaetaan eriin, ja jokainen erä syötetään ensin diskriminaattorille ja sitten generaattorille. Diskriminaattorin koulutusvaiheessa sille syötetään sekä aitoja korkealaatuisia ääninäytteitä että generaattorin tuottamia väärennöksiä. Diskriminaattorin tavoitteena on oppia erottamaan nämä kaksi toisistaan. Generaattorin koulutusvai-

heessa sen tavoitteena on tuottaa niin vakuuttavia väärennöksiä, että diskriminaattori ei pysty niitä erottamaan aidoista.

Koulutusskripti sisältää erilliset funktiot diskriminaattorin ja generaattorin koulutusaskelille. Nämä funktiot laskevat häviöt, suorittavat backpropagationin ja päivittävät mallien painot optimoijien avulla, kuten Adam. Generaattorin koulutuksessa hyödynnetään myös L1-häviötä Mel-spektrogrammien välillä, mikä auttaa generaattoria tuottamaan ääniä, jotka ovat spektraalisesti lähempänä aitoja näytteitä.

Koulutusprosessin aikana malleista ja tuotetuista ääninäytteistä otetaan säännöllisesti tallenteita. Tämä mahdollistaa koulutuksen edistymisen seuraamisen ja parhaiden mallien säilyttämisen. Koulutusskripti tallentaa sekä väliaikaisia malleja että säännöllisin väliajoin tuotettuja ääninäytteitä eri vaiheista. Lisäksi koulutuksen lopussa tallennetaan lopulliset mallit.

3.4 Käytetyt mittarit äänenlaadun arviointiin

Erilaiset mittarit ovat tärkeitä äänenlaadun arvioinnissa, ja niitä käytetään monissa äänenkäsittelyprojekteissa, erityisesti silloin, kun pyritään parantamaan puheen tai musiikin laatua. Tässä projektissa käytetään kahta yleisesti käytettyä menetelmää äänenlaadun arvioinnissa: Mel-Frequency Cepstral Coefficients (MFCC) ja ihmisarviointi.

Mel-Frequency Cepstral Coefficients, eli MFCC on yksi suosituimmista piirreerottelumenetelmistä äänenkäsittelyssä. Se perustuu mel-asteikkoon, joka jäljittelee ihmisen kuulon havaintoja taajuuksista. MFCC:n avulla voidaan tehokkaasti analysoida ja esittää äänen taajuuspiirteitä. Ne ovat erityisen hyödyllisiä puheentunnistuksessa, musiikin analyysissä ja äänenlaadun arvioimisessa. Ne tarjoavat yksityiskohtaisen esityksen äänen taajuuspiirteistä, mikä tekee niistä tehokkaita. (OpenGenus IQ, 2023)

Ihmisarviointi perustuu kuuntelijoiden subjektiivisiin havaintoihin ja arvioihin äänen laadusta. Ihmisarviointi on erityisen hyödyllinen, koska se ottaa huomioon ihmisen kuulon ja kognitiiviset prosessit, joita objektiiviset menetelmät eivät aina pysty täysin mallintamaan. Se on erityisen hyödyllinen äänenlaadun subjektiivisten piirteiden arvioinnissa, kuten äänen luonnollisuuden, selkeyden ja miellyttävyyden arvioinnissa. Se on myös tärkeä osa äänihäiriöiden diagnosointia ja hoidon tehokkuuden arviointia.

4 TOTEUTUS

4.1 Ohjelmointiympäristö ja työkalut

Ohjelmistokehitys on moniulotteinen prosessi, joka vaatii paitsi ymmärrystä itse koodista, myös asianmukaisen työympäristön ja tehokkaiden työkalujen hyödyntämistä. Ohjelmointiympäristö muodostaa sen kehyksen, jonka sisällä kehitystyötä tehdään, kun taas työkalut ovat ne spesifiset sovellukset ja kirjastot, jotka mahdollistavat koodin kirjoittamisen, testaamisen, suorittamisen ja hallinnoinnin. Tämä projekti käyttää Python 3.13 ohjelmointikieltä, Pythonin sisäänrakennettua venv virtuaaliympäristöä sekä PyTorch-kirjastoa (torch ja torchaudio).

Python on noussut vuosien varrella yhdeksi maailman suosituimmista ohjelmointikielistä, eikä syyttä. Sen luettavuus, monipuolisuus ja laaja kirjastovalikoima tekevät siitä erinomaisen valinnan hyvin monenlaisiin tehtäviin, varsinkin koneoppimiseen.

Projekti sisältää erilaisia riippuvuuksia alla pyörivään järjestelmään verrattuna, joten tähän yleiseen ongelmaan ratkaisun tarjoaa virtuaaliympäristö. Pythonin sisäänrakennettu venv-moduuli on kevyt ja tehokas tapa luoda eristettyjä Python-ympäristöjä. Kun projekti on aktivoitu venv-ympäristössään, kaikki asennetut kirjastot sijaitsevat vain kyseisessä ympäristössä, eivätkä ne sotke tai vaikuta järjestelmän globaaliin Python-asennukseen tai muihin virtuaaliympäristöihin. Tämä eristys on ensiarvoisen tärkeää riippuvuuksien hallinnassa, yhteensopivuusongelmien välttämisessä ja projektin toistettavuuden varmistamisessa. Kehittäjä voi luottaa siihen, että projektin riippuvuudet pysyvät vakioina ja hallinnassa, riippumatta muista koneen projekteista.

PyTorch on avoimen lähdekoodin koneoppimiskirjasto, joka tunnetaan joustavuudestaan ja helppokäyttöisyydestään erityisesti syväoppimismallien

rakentamisessa ja koulutuksessa. PyTorchin ydin (torch) tarjoaa tehokkaan tensorilaskennan ja automaattisen derivoinnin tuen, mikä on kriittistä neuroverkkojen optimoinnissa. Tämän lisäksi PyTorch-ekosysteemiin kuuluu erikoistuneita kirjastoja, kuten torchaudio, joka on suunniteltu erityisesti äänidatan käsittelyyn ja analysointiin, tarjoten työkaluja esimerkiksi äänisignaalien esikäsittelyyn ja syväoppimismallien kehittämiseen äänitehtäviin. PyTorchin vahva tuki GPU-kiihdytykselle tekee siitä myös erinomaisen valinnan laskennallisesti raskaiden tehtävien suorittamiseen.

Yhdistämällä nämä komponentit luodaan tehokas ja hallittu kehitysympäristö, joka sopii loistavasti tämän projektin tarpeisiin. Python 3.13 toimii alustana, tarjoten modernin ja tehokkaan ohjelmointikielen. venv-virtuaaliympäristö varmistaa, että PyTorch-kirjaston ja sen riippuvuuksien (kuten torch ja torchaudio) asennus on eristetty ja projektispesifi. Näin voidaan varmistaa toistettavuus myös muilla järjestelmillä ja tarvittaessa myös Docker-kontitus. (Ouro 2022)

4.2 Neuroverkon arkkitehtuuri ja parametrit

Projektin Python-koodi sisältää kaksi pääkomponenttia: generator.py ja discriminator.py. Nämä skriptit määrittelevät generaattorin ja diskriminaattorin.

Generaattori (SISUGenerator)

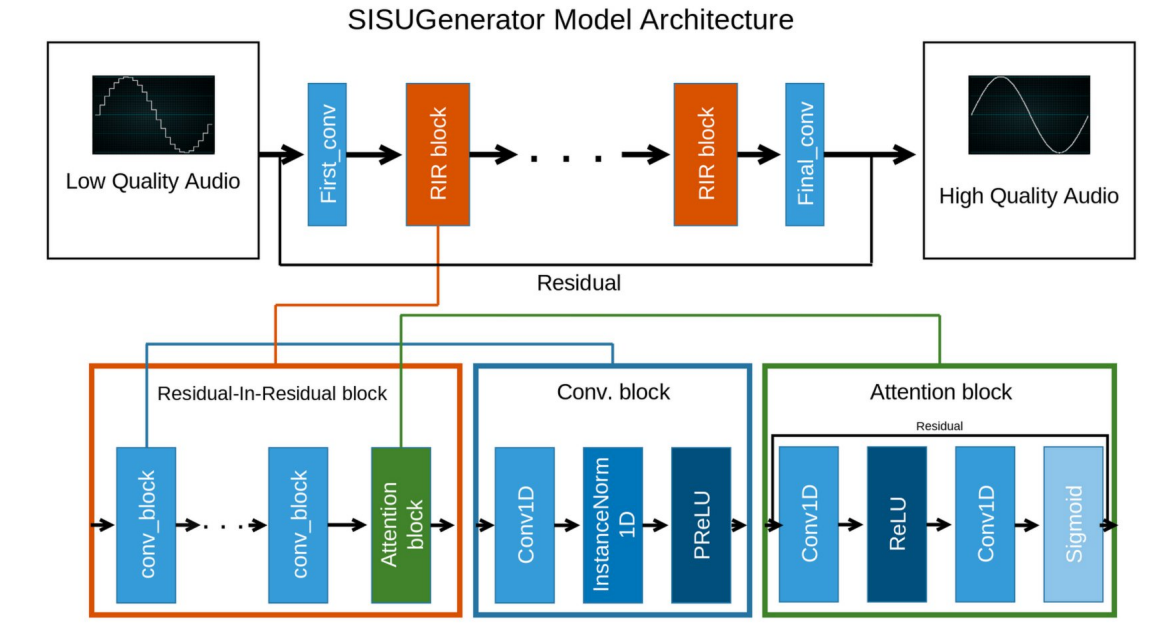
SISUGenerator on suunniteltu käsittelemään yksiulotteista signaalia ja luomaan korjaussignaalin heikkolaatuiselle signaalille.

Generaattorin päärakenne:

1. Alkukonvoluutio (self.conv1): Sisään tuleva monoääni konvoloidaan useammaksi kanavaksi (channels). Tämä alustuskerros käyttää suhteellisen suurta kerneliä (koko 7) ja sopivaa paddingia ajallisen ulottuvuuden säilyttämiseksi. Normalisointiin käytetään InstanceNorm1d ja aktivointifunk-

tiona PReLU. Tämä kerros poimii alustavia ominaisuuksia raa'asta ääniaallosta.

2. Residual-in-Residual -lohkoketju (self.rir_blocks): Tämä on generaattorin ydin. Ketju koostuu useista ResidualInResidualBlock-lohkoista. Nämä lohkot sisältävät useita peräkkäisiä conv_blockeja sekä yhden Attention-Blockin. Tärkeintä on residuaaliyhteys: lohkon ulostulo on sen sisäisen prosessoinnin (konvoluutiot + attention) ja lohkon sisäänmenon summa. Tämä rakenne helpottaa gradienttien kulkua ja auttaa verkkoa oppimaan "korjauksia" tai "lisäyksiä" signaaliin alkuperäisen signaalin sijaan.
3. Seuraavana on konvoluutiolohkot, jotka ovat suunniteltuja käsittelemään yksiulotteista äänsignaalia. Ne suorittavat kolme vaihetta peräkkäin: ensin yksiulotteinen konvoluutio etsii signaalista piirteitä tai kuvioita, sitten normalisointi säätää arvoja tasoittaakseen signaalin voimakkuusvaihteluita ja helpottaakseen oppimista, ja lopuksi aktivointi lisää epälineaarisuutta, jotta verkko voi oppia monimutkaisempia riippuvuuksia. Tärkeää on, että täytön (paddingin) (signaalin alkuun ja loppuun lisättyjen arvojen) ansiosta tämä koko prosessi ei muuta signaalin alkuperäistä pituutta.
4. AttentionBlock: Yksinkertainen kanavahuomiomekanismi. Se oppii painottamaan eri kanavia niiden tärkeyden perusteella, todennäköisesti auttaen verkkoa keskittymään informatiivisimpiin piirteisiin äänessä.
5. Loppuun lisättävä residuaali (learned_residual): RIRB-lohkoketjun jälkeen final_layer (1D-konvoluutio, joka palauttaa kanavamäärän takaisin yhteen) laskee signaalin, jota kutsuin koodissa learned_residualiksi.
6. Generaattorin lopullinen ulostulo saadaan lisäämällä tämä opittu residuaali alkuperäiseen syötteeseen (residual_input) skaalattuna parametrilla alpha. Generaattori siis yrittää oppia tuottamaan "korjaussignaalia". Alpha-parametrilla voidaan säätää, kuinka voimakkaasti opittu korjaus vaikuttaa lopulliseen ulostuloon.



KUVA 2. SISUGenerator-generaattorin arkkitehtuuri

Diskriminaattori (SISUDiscriminator)

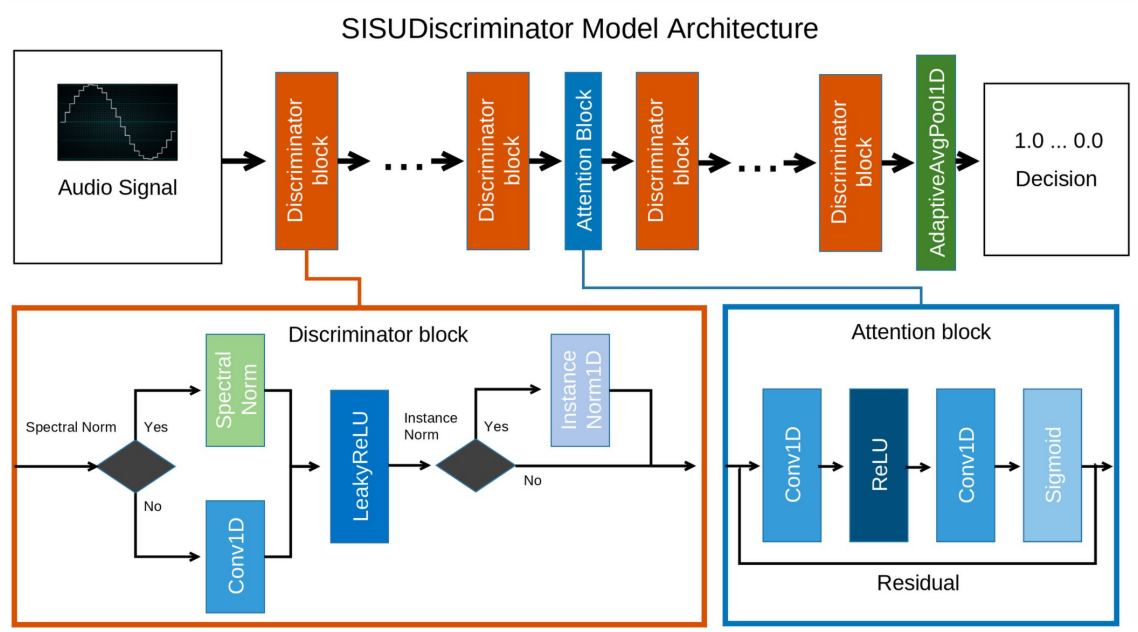
SISUDiskriminaattorin tehtävä on erottaa todelliset, korkealaatuiset äänisignaalit generaattorin parannetuista signaaleista.

Diskriminaattorin päärakenne:

1. Konvoluutioiden ja alaspäinnäytteistyksen ketju (self.model): Diskriminaattori koostuu peräkkäisistä discriminator_block-lohkoista.
2. Discriminator_block on diskriminaattorin perusrakennuspalikka, joka käsittelee yksiulotteista signaalia monipuolisella konvoluutiolla, lisää epälineaarisuutta LeakyReLU:lla ja käyttää normalisointeja (erityisesti spectral normia) koulutuksen vakauttamiseksi, auttaen erottamaan aidon ja väärennetyn datan.
3. AttentionBlock: Myös diskriminaattori sisältää kanavahuomiomekanismin. Tämä auttaa diskriminaattoria keskittymään niihin kanaviin, jotka sisältävät tärkeimpiä piirteitä erottelua varten.
4. Ensimmäinen ja viimeinen discriminator_block eivät käytä InstanceNorm1d -normalisointia. Viimeinen lohko ei myöskään käytä spektraalinormi-

tusta ja palauttaa ulos yhden kanavan, joka toimii "reaalisuus" -pisteytyksen perustana.

5. Globaali keskiarvopoolaus: Konvoluutiokerrosten ketjun jälkeen käytetään AdaptiveAvgPool1d-poolausta. Tämä laskee kaikkien ajallisten pisteiden keskiarvon kullekin kanavalle, tiivistäen koko signaalin ominaisuudet yhteen vektoriin.
6. Mallin luoma vektori muunnetaan yhdeksi skalaariksi. Tämä arvo edustaa diskriminaattorin arviota siitä, kuinka "todellinen" tai "korkealaatuinen" annettu äänisignaali on.



KUVA 3. SISUDiscriminator-diskriminaattorin arkkitehtuuri

4.3 Äänidatan hankinta ja valmistelu

Datanjalostusprosessia toteutetaan projektin data.py-tiedoston AudioDataset-luokassa. Tässä luokassa äänidata ladataan ja valmistellaan koulutusta varten. Keskeinen osa valmistelua on luoda pareja korkealaatuisesta ja keinotekoisesti heikennetystä äänestä.

Datan lataaminen ja perusrakenne

AudioDataset-luokka on suunniteltu toimimaan DataLoaderin kanssa, joka vastaa datan puolittamisesta ja rinnakkaisesta lataamisesta. Luokan konstruktori (__init__) ottaa vastaan syötekansion (input_dir) ja laitetiedon (device, esim. 'cpu' tai 'cuda'). Konstruktori etsii rekursiivisesti kaikki .wav-päätteiset tiedostot annetusta syötekansiosta ja tallentaa niiden polut listaan self.input_files. Tämä lista muodostaa perustan datasetille, ja luokan __len__-metodi palauttaa tämän listan pituuden, eli datasetin sisältämien äänitiedostojen määrän.

Tärkein metodi on __getitem__(idx), joka vastaa yhden yksittäisen ääninäytteen käsittelystä. Se ottaa indeksin idx ja palauttaa parin, joka koostuu käsitellystä korkealaatuisesta ja heikkolaatuisesta äänestä.

Äänen laadun heikentäminen

Yksi keskeisimmistä vaiheista __getitem__-metodissa on heikkolaatuisen äänen luominen alkuperäisestä, korkealaatuisesta äänestä. Tämä simuloi usein erilaisia äänenlaadun heikkenemisen syitä, kuten häviöllistä pakkausta tai alhaisella näytteenottotaajuudella tallentamista.

Prosessi alkaa lataamalla alkuperäinen äänitiedosto torchaudio.load-funktiolla. Tällöin saadaan äänidata tensorina (high_quality_audio) ja sen alkuperäinen näytteenottotaajuus (original_sample_rate).

Seuraavaksi valitaan satunnaisesti yksi näytteenottotaajuus self.audio_sample_rates-listasta (mangled_sample_rate). Ääni alasnäytteistetään alkuperäisestä näytteenottotaajuudesta tähän valittuun "heikennettyyn" näytteenottotaajuuteen käyttäen torchaudio.transforms.Resample-transformaatiota.

Tämän jälkeen heikennetty ääni ylössäytteistetään takaisin alkuperäiseen näytteenottotaajuuteen. Tämä kaksivaiheinen prosessi (alasnäytteistys ja sitten

ylössäytteistys) on tyypillinen tapa simuloida äänenlaadun heikkenemistä. Pelkkä alasnäytteistys vähentäisi datan kokoa, mutta ylössäytteistys takaisin alkuperäiseen taajuuteen varmistaa, että heikennetty ja alkuperäinen ääni voidaan syöttää mallille samassa muodossa (samalla näytteenottotaajuudella), mutta heikennetty versio sisältää alasnäytteistyksen aiheuttamia vääristymiä ja informaation menetystä.

Lisäkäsittely ja standardointi

Äänenlaadun heikentämisen jälkeen molemmat äänitensorit (alkuperäinen korkealaatuinen ja keinotekoisesti heikennetty) muunnetaan yksikanavaiseksi monoääneksi käyttäen erillistä AudioUtils.stereo_tensor_to_mono-funktiota.

Tärkeä standardointivaihe on ääninäytteiden pituuden normalisointi arvoon self.MAX_LENGTH. MAX_LENGTH on asetettu arvoon 44100. Jos alkuperäinen näytteenottotaajuus on 44100 Hz (yleinen CD-laatu), tämä tarkoittaa yhden sekunnin pituista ääninäytettä. Molemmat äänitensorit tarkistetaan:

1. Jos tensorin pituus on lyhyempi kuin MAX_LENGTH, se täytetään nollilla (tai ’pädätään’ epävirallisesti) loppuun MAX_LENGTH-pituiseksi käyttäen torch.nn.functional.pad (lyhennetty F.pad).
2. Jos tensorin pituus on pidempi kuin MAX_LENGTH, se katkaistaan MAX_LENGTH-pituiseksi alusta.

Tämä paddingi/leikkaus suoritetaan sekä korkealaatuiselle että heikkolaatuiselle äänelle erikseen, mutta lopputuloksena molemmilla on sama standardoitu pituus. Tämä yhtenäinen pituus on välttämätön edellytys, jotta ääninäytteet voidaan koota eriin (batch) mallille syötettäväksi.

Lopuksi molemmat käsitellyt äänitensorit siirretään määritellylle laitteelle (self.device), olipa se sitten CPU tai GPU, valmisteluna mallin syötteeksi. __getitem__-metodi palauttaa sitten parin, joka sisältää korkealaatuisen äänen tensorin ja alkuperäisen näytteenottotaajuuden sekä heikkolaatuisen äänen tensorin ja sen heikentämiseen käytetyn näytteenottotaajuuden.

4.4 Koulutus- ja testausmenetelmät

Koulutusprosessi alkaa datan lataamisella ja valmistelulla. AudioDataset-luokkaa, jonka toimintaa kuvattiin aiemmin, käytetään lukemaan äänitiedostot määritellystä syötekansiosta. Tämä luokka luo jokaisesta alkuperäisestä äänitiedostosta parin: alkuperäisen (korkealaatuisen) ja keinotekoisesti heikennetyn (heikkolaatuisen) version. Kuten aiemmin mainittiin, tietoaineistossa varmistetaan myös, että kaikki ääninäytteet ovat samanpituisia (määritelty MAX_LENGTH-arvolla) ja että ne ovat muutettu yksikanavaiseksi.

DataLoader sitten käärii tämän datasetin ja vastaa datan lataamisesta erissä (batch) ja niiden sekoittamisesta joka epookilla. Datan lataus tapahtuu taustalla, mikä tehostaa koulutusprosessia.

Mallien ja optimointimekanismien alustus

Koulutus-skripti alustaa kaksi päämallia: SISUGenerator ja SISUDiscriminator. Nämä ovat GAN-arkkitehtuurin kaksi osaa. Koodi mahdollistaa olemassa olevien, aiemmin koulutettujen mallien lataamisen joko tiettyjen polkujen perusteella tai väliaikaisista tiedostoista (temp_generator.pt, temp_discriminator.pt) koulutuksen jatkamiseksi. Mallit siirretään valitulle laskentalaitteelle (device, tyypillisesti GPU, jos saatavilla).

Koulutus vaatii häviöfunktiot ja optimoijat. Molemmille malleille (Generaattori ja Diskriminaattori) käytetään nn.BCEWithLogitsLoss-häviöfunktiota, joka on tyypillinen valinta GAN-verkoissa, joissa ennustetaan binääristä luokittelua (aito vs. väärennös). Optimoijina toimivat optim.Adam-algoritmit. Lisäksi käytössä on oppimisnopeuden ajastus (scheduler), ReduceLROnPlateau, joka laskee oppimisnopeutta, jos häviöfunktio ei parane tietyn kärsivällisyysjakson aikana.

Koulutussilmukka

Itse koulutus tapahtuu start_training()-funktion sisällä. Koulutussilmukka koostuu ulommasta silmukasta (epookit) ja sisemmästä silmukasta (erät).

Jokaisen erän käsittelyssä tapahtuu tyypillinen GAN-koulutuskierto: ensin koulutetaan Diskriminaattoria, sitten Generaattoria.

Diskriminaattorin koulutus:

1. Diskriminaattori asetetaan koulutustilaan (discriminator.train()).
2. Koulutusfunktiota kutsutaan, ja sille annetaan erä aitoja (korkealaatuisia) ääninäytteitä ja vastaava erä heikkolaatuisia ääninäytteitä.
3. Generaattoria käytetään tuottamaan parannettuja ääninäytteitä näistä heikkolaatuisista näytteistä (tässä vaiheessa Generaattoria ei kouluteta, vain sen tuotosta käytetään kouluttamiseen).
4. Diskriminaattori yrittää oppia erottamaan aidot korkealaatuiset äänet (merkitään numerolla 1) generaattorin tuottamista parannetuista äänistä (merkitään numerolla 0).
5. Diskriminaattorin häviö lasketaan ja sen painot päivitetään optimizer_d:n avulla.

Generaattorin koulutus:

1. Generaattori asetetaan koulutustilaan (generator.train()).
2. Koulutusfunktiota kutsutaan, ja sille annetaan erä heikkolaatuisia ääninäytteitä, vastaava erä aitoja (korkealaatuisia) ääninäytteitä sekä mel_transform, stft_transform ja mfcc_transform mittaamaan tuotetun äänen laatua.
3. Generaattori tuottaa parannetut ääninäytteet heikkolaatuisista äänistä ja ne syötetään diskriminaattorille (tässä vaiheessa Diskriminaattoria ei kouluteta, vain sen ennustusta käytetään).
4. Generaattoria koulutetaan siten, että Diskriminaattori erehtyisi ja luokittelisi generaattorin tuotoksen aidoksi. Tämä on kilpaileva häviö (englanniksi adversarial loss).

5. Koulutusfunktio laskee häviöitä, jotka perustuvat spektrogrammiin, Mel-spektrogrammiin ja MFCC-piirteisiin. Nämä ovat ns. "perceptual losses" tai "content losses", jotka ohjaavat Generaattoria tuottamaan parannettua ääntä, joka ei ainoastaan huijaa Diskriminaattoria, vaan myös kuulostaa tai näyttää piirteidensä puolesta aidolta korkealaatuiselta ääneltä. Generaattorin kokonaishäviö on yhdistelmä adversarial lossia ja näitä muita häviöitä.
6. Generaattorin kokonaishäviö lasketaan ja sen painot päivitetään optimizer_g:n avulla.

Kummankin mallin optimoijat ajastetaan (scheduler_d.step, scheduler_g.step) kunkin koulutuserän jälkeen, perustuen häviöihin.

Edistymisen seuranta ja tallennus

Malleja ja esimerkkiääniä tallennetaan säännöllisesti. Jokaisen epookin jälkeen (tai tietyin välein, projektissa joka 25. epookki) tallennetaan yksi esimerkki heikkolaatuisesta, Generaattorin parantamasta ja alkuperäisestä korkealaatuisesta ääninäytteestä WAV-muotoon torchaudio.save-funktiolla. Tämä mahdollistaa parannusprosessin kuuntelemisen ja laadun visuaalisen seurannan.

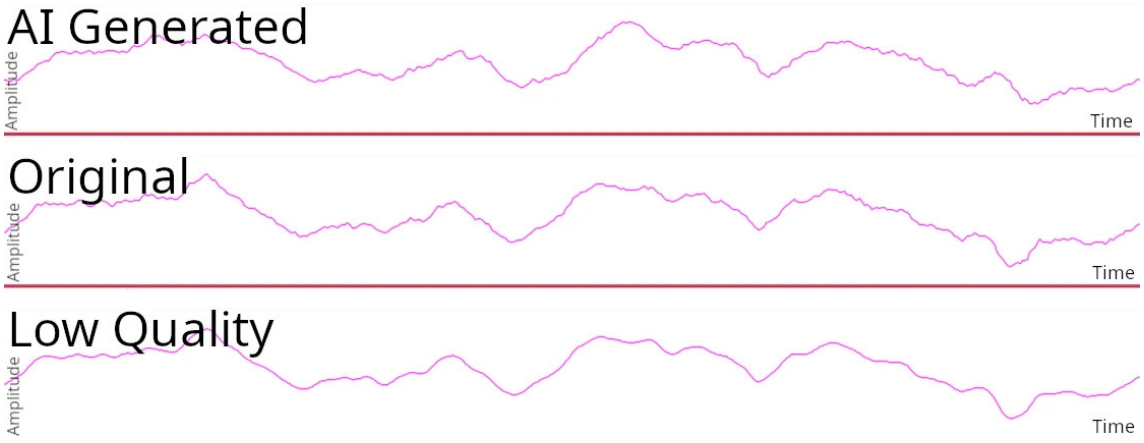
Malleista tallennetaan säännöllisesti väliaikaiset versiot (temp_discriminator.pt, temp_generator.pt) ja nykyinen epookkinumero (epoch_data.json). Tämä on tärkeää koulutuksen jatkamiseksi keskeytyksen jälkeen. Koulutuksen loputtua (määritellyn epookkimäärän jälkeen) tallennetaan lopulliset mallitiedostot.

5 TULOKSET

5.1 Äänenlaadun parannuksen tulokset

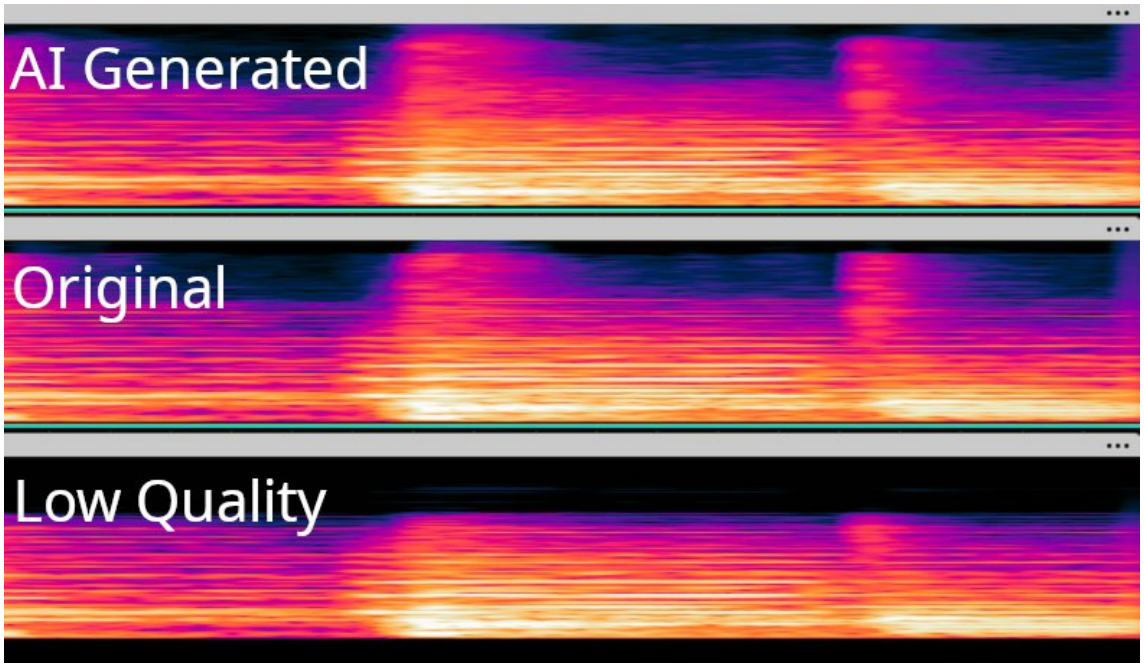
Saavutetut tulokset tässä projektissa ovat hyvin lupaavia. Generaattorimallia on koulutettu 1 440 sukupolven ajan yli 2 500 ääninäytteellä ja malli onnistuu jo tässä vaiheessa luomaan yksityiskohtia heikkolaatuiseen ääneen siten, että se muistuttaa alkuperäistä hyvälaatuista ääntä. Mallin kyky "kuvitella" ja luoda uskottavaa sisältöä tälle puuttuvalle alueelle on osoitus sen oppimiskyvystä ja lähestymistavan potentiaalista. Suurin osa heikkolaatuisesta äänestä on parantunut merkittävästi, mutta tuotetut ääninäytteet sisältävät pientä kohinaa.

Kuvassa nähdään ääniraidat tekoälyn parantamalle, alkuperäiselle ja keinotekoisesti heikennetylle äänelle. Heikon äänen ääniraita on hyvin pehmeä verrattuna parannettuun ja alkuperäiseen. Kuvaajasta voidaan huomata, että alkuperäisessä ja tekoälyn parantamassa ääniraidassa on samankaltaista rosoisuutta. Tämä rosoisuus kuuluu selvästi musiikissa; esimerkiksi trumpetit ja muut vaskipuhaltimet kuulostavat eloisammalta, toisin kuin heikennetyssä äänessä. Generaattori tuo kuitenkin ääneen ylimääräistä värähtelyä, joka havaitaan kohinana.



KUVA 4. Tekoälyn parantaman, alkuperäisen ja huonolaatuisen äänen ääniraidat

Spektogrammikuvassa nähdään korkeiden taajuuksien kohdalla selviä eroja. Alkuperäisessä korkea ääni on tarkempaa ja se havaitaan ylhäällä vaakatasossa olevina piikkeinä. Tekoälyn parantamassa äänessä on myös nämä piikit, mutta ne ovat sumeita. Heikkolaatuinen ääni taas ei sisällä piikkejä.



KUVA 5. Ääniraitojen spektogramminäkymä

5.2 Tulosten analyysi ja tulkinta

Äänen parantaminen 11025 Hz:stä 44100 Hz:iin on osoittautunut lupaavaksi, sillä merkittävä osa alkuperäisestä laadusta ja yksityiskohdista on pystytty palauttamaan. Tutkimalla jokaisen ääninäytteen ääniraitaa huomasin, että heikennetyn ja alkuperäisen ääniraidan välillä on vielä puutteita, kuten tutkimus tässä vaiheessa on osoittanut. Mallin tuottama ääninäyte on heikennetyn ja alkuperäisen ääniraidan välimaastossa eli tekoälyn tuottama ääni sisältää molempien ääninäytteiden sisältämiä yksityiskohtia, mutta lisätyllä kohinalla. Tämä kohina on peräisin generaattorista, jota pitää vielä opettaa laajemmalla tietoaineistolla (englanniksi dataset) kauemmin.

5.3 Mahdolliset ongelmat ja niiden ratkaisut

Jäljelle jäävä kohinan poisto on kuitenkin selkeä kehityskohde, ja sen vähentäminen vaatii mallin lisäkouluttamista ja jatkotutkimusta arkkitehtuurin hienosäätämiseksi, häviöfunktioiden painotuksen optimoimiseksi tai mahdollisesti erillisten kohinanvaimennustekniikoiden integroimiseksi parannusprosessiin. Tulokset ovat kuitenkin vahva osoitus siitä, että GAN-pohjaiset syväoppimismallit pystyvät tehokkaasti käsittelemään ja parantamaan vakavasti heikennettyä äänidataa.

6 POHDINTA

6.1 Työn onnistuminen ja haasteet

Projektin onnistumisen kirkkain osoitus ovat kuvatut tulokset äänenlaadun parannuksessa 11025 Hz:stä 44100 Hz:iin. Ottaen huomioon, että äänenlaadun palauttaminen tällaisesta rajusta heikennyksestä on erittäin vaikeaa – puuttuva korkeataajuinen informaatio täytyy käytännössä syntetisoida uskottavasti. Malli on kyennyt palauttamaan merkittävästi äänen selkeyttä, yksityiskohtia ja dynamiikkaa. Se on onnistunut luomaan sisältöä taajuusalueille, jotka olivat alkuperäisestä heikennetystä äänestä poissa.

Mallin kouluttaminen ja arkkitehtuurin suunnittelu on ollut haastavaa. Pieni määrä tasoja vaikuttaa suuresti tuotettuun ääneen ja liian suuri määrä tekee koulutuksesta paljon raskaampaa. Suurin haaste on ollut sopivan arkkitehtuurin löytämisessä, konvoluutioverkon tasojen määrän laatimisessa ja tarvittavien huomiomekanismien asettamisessa.

Kuitenkin tuotetut näytteet sisältävät vähän kohinaa. Vaikka projekti on erittäin onnistunut perustehtävässään, on vielä varaa hienosäätöön ja parantamiseen. Kohina on usein merkki siitä, että malli ei pysty täydellisesti erottamaan signaalia ja ei-toivottuja artefakteja synteesiprosessissa. Se voi johtua nykyisien häviöfunktioiden kyvystä rankaista tämän tyyppisestä epäpuhtaudesta. Tämä on selkeä seuraava kehityskohde.

6.2 Tulosten yleistettävyys

Malli toimii hyvin opetusdatassa olevilla näytteillä, ja se kykenee myös parantamaan laadukkaasti sellaista heikennettyä ääntä, jota se ei ole nähnyt koulutusvaiheen aikana. Nämä ääninäytteet ovat kuitenkin samasta lähteestä kuin opetusdatan ääninäytteet, joten uuden musiikkigenren ääninäytteiden parantaminen on vielä este tälle mallille.

6.3 Jatkotutkimusideat

Nykyinen äänenlaadun parannusmalli on koulutettu käsittelemään kiinteän mittaisia, suhteellisen pitkiä ääninäytteitä. Monet potentiaaliset ja kiinnostavat käyttökohteet, kuten reaaliaikainen viestintä (videoneuvottelut, puhelut), suoratoistopalvelut tai live-ääniefektit, vaativat kuitenkin käsittelyä reaaliajassa – toisin sanoen, ääntä on käsiteltävä sitä mukaa kuin sitä saapuu, mahdollisimman pienellä viiveellä (latenssilla). Tämän vuoksi mallin muuttaminen reaaliaikaiseksi on luonnollinen ja arvokas jatkotutkimussuunta.

Keskeisin ero normaalin ja realiaikaisen välillä on tarve käsitellä jatkuvaa äänivirtaa pienissä, peräkkäisissä paloissa hyvin lyhyellä latenssilla. Malli, joka on koulutettu yhden sekunnin pätkiin, ei välttämättä toimi sellaisenaan esimerkiksi vain 64 millisekunnin (noin 2822 näytettä 44.1 kHz:llä) mittaisilla syötteillä.

Puskurointi on välttämätöntä, jotta jatkuvasta sisääntulevasta äänivirrasta saadaan kerättyä riittävän pitkä pätkä kerrallaan mallille syötettäväksi. Ääni saapuu järjestelmään tyypillisesti pieninä paloina, ja puskuri kerää näitä paloja, kunnes se on riittävän täynnä muodostamaan yhden käsiteltävän pätkän (esim. 64 ms). Tämä puskuri tyhjennetään tai sen sisältöä siirretään käsiteltäväksi sopivin väliajoin, ja samalla uutta ääntä kerätään puskurin loppuun.

7 YHTEENVETO

7.1 Työn päätulokset ja niiden merkitys

Projektin päätulos on merkittävän äänenlaadun parannuksen saavuttaminen erittäin heikosta lähtökohdasta. Se validoi käytetyn metodologian tehokkuuden monimutkaisessa signaalirestaurointitehtävässä, avaa potentiaalia käytännön sovelluksiin ja tarjoaa vankan pohjan tulevalle tutkimukselle mallin suorituskyvyn ja käytettävyyden parantamiseksi. Vaikka täydellisyyteen ei vielä ole päästy, saavutettu parannuksen aste on selvä osoitus projektin onnistuneisuudesta ja sen potentiaalista vaikuttaa positiivisesti äänidatan käsittelyyn ja sen laatuun tulevaisuudessa.

7.2 Työn vaikutus äänenlaadun parantamisen alalla

Projekti vahvistaa entisestään syväoppimisen asemaa ja potentiaalia äänenkäsittelyn alalla. Vain muutama vuosi sitten tällaisen monimutkaisten signaalimuunnosten, erityisesti informaation synteesin ja laadun palautuksen, toteuttaminen neuroverkoilla oli haastavaa. Tämä työ osoittaa konkreettisesti, että nykyaikaiset syväoppimismallit, kuten tässä käytetty GAN, kykenevät oppimaan erittäin monimutkaisia, epälineaarisia kuvauksia äänisignaalien välillä.

Työ tekee merkittävän panoksen erityisesti vakavasti heikentyneen äänen käsittelyn ja kaistanlaajennuksen alalohkoon. Monet äänenparannusmenetelmät keskittyvät tyypillisesti lievän kohinan poistoon, kaikujen vaimennukseen tai kohtuullisen kompression aiheuttamien artefaktien lieventämiseen. Sen sijaan äänen palauttaminen tilanteessa, jossa merkittävä osa taajuusspektristä (esim. kaikki yli 5.5 kHz:n taajuudet 11 kHz -> 44 kHz tapauksessa) on kadonnut, on paljon radikaalimpi ja teknisesti vaikeampi ongelma. Projektin saavuttamat tulokset osoittavat, että syväoppiminen voi tarjota tehokkaan ratkaisun juuri tähän äärimmäisen vaativaan tehtävään, jota perinteisillä menetelmillä on ollut vaikea saavuttaa uskottavasti.

Projekti toimii konkreettisena validointina tietynlaisille metodologisille valinnoille. GAN-arkkitehtuurin käyttö yhdessä todennäköisesti käytettyjen spektriin tai äänen piirteisiin perustuvien "perceptual losses" -häviöfunktioiden kanssa osoittautuu tehokkaaksi yhdistelmäksi tämänkaltaisessa synteesiä ja laadun parannusta yhdistävässä tehtävässä. Projektin onnistuminen tämän lähestymistavan avulla antaa muille alan tutkijoille vahvoja perusteita harkita ja edelleen kehittää vastaavia mallirakenteita ja häviöfunktioiden yhdistelmiä myös muihin samankaltaisiin äänenkäsittelyn ongelmiin.

Samalla kun työ osoittaa merkittäviä saavutuksia, se myös kirkastaa ja määrittelee alan jäljellä olevia haasteita. Maininta jäljelle jäävästä kohinasta ja tarve varmistaa tulosten yleistettävyys opetusdatan ulkopuolelle ovat tärkeitä havaintoja. Ne osoittavat muille tutkijoille, mitkä ovat tämänkaltaisten mallien nykyiset rajoitukset ja minne kehitysresursseja tulisi seuraavaksi suunnata – esimerkiksi tehokkaampien kohinanvaimennusmekanismien integrointiin tai parempien yleistettävyyden varmistavien koulutusstrategioiden kehittämiseen.

Projektin osoittama kyky palauttaa merkittävästi laatua erittäin heikosta äänestä voi inspiroida uusia käytännön sovelluksia. Monet historialliset äänitykset, matalan bittivirran viestintä tai ääni haastavissa ympäristöissä kärsivät vakavasta laadun heikkenemisestä. Tämän työn kaltaisten mallien potentiaali tehdä tällaisesta äänestä ymmärrettävämpää tai miellyttävämpää voi johtaa uusien ohjelmistojen, laitteiden tai palveluiden kehittämiseen arkistoinnissa, viestintätekniikassa tai äänen analyysissä.

Yhteenvetona voidaan todeta, että tämä projekti on tärkeä lisä äänenlaadun parantamisen alalle. Se vahvistaa syväoppimisen roolia monimutkaisissa signaalitehtävissä, edistää erityisesti vakavasti heikentyneen äänen käsittelyä, validoi tehokkaita metodologisia lähestymistapoja, ohjaa tulevaa tutkimusta tunnistamalla keskeisiä haasteita ja inspiroi potentiaalisia uusia sovelluksia. Tulokset osoittavat, että merkittävienkin laadunmenetysten palauttaminen syväoppimisen avulla on realistinen tavoite.

LÄHTEET

Abdul, Z. K., Al-Talabani, A. K., 2022, Mel Frequency Cepstral Coefficient and its Applications: A Review. IEEE Access, Vol. 10, pp. 122136-122158. Pdf-tiedosto. Saatavilla: https://doi.org/10.1109/ACCESS.2022.3223444 Luettu: 12.5.2025.

Durak, L., Arikan, O., 2003, Short-time Fourier transform: two fundamental properties and an optimal implementation. IEEE Transactions on Signal Processing. Pdf-tiedosto. Saatavilla: https://doi.org/10.1109/TSP.2003.810293 Luettu: 12.5.2025.

Federal Agencies Digitization Guidelines Initiative, 2025, Sampling Rate (Audio), Digitization Guidelines. Saatavilla osoitteessa: https://www.digitizationguidelines.gov/term.php?term=samplingrateaudio Luettu: 16.5.2025

Goodfellow, I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., 2014. Generative Adversarial Nets. Pdf-tiedosto. arxiv.org. Luettavissa: https://doi.org/10.48550/arXiv.1406.2661 Luettu: 15.5.2025

Kingma, D. P., Ba, J., 30.1.2017, Adam: A Method for Stochastic Optimization. Pdf-tiedosto. arxiv.org. Luettavissa: https://doi.org/10.48550/arXiv.1412.6980 Luettu: 9.5.2025

Hadesty, L., 14.4.2017, Explained: Neural networks and deep learning, MIT News. Luettavissa: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 Luettu: 12.5.2025

OpenGenus IQ, 2023, MFCC (Mel Frequency Cepstral Coefficient) in Audio Processing. Saatavilla: https://iq.opengenus.org/mfcc-audio Luettu: 17.4.2025

Ouro, 2022, Mitä ovat kontit ja Docker?, Luettavissa: https://ouro.fi/mita-ovat-kontit-ja-docker Luettu: 28.4.2025

Papers With Code, 2025, Leaky ReLU. Saatavilla: https://paperswithcode.com/method/leaky-relu Luettu: 9.5.2025

Phonak, 2025, Tekoälykuulokojeet. Saatavilla: https://www.phonak.com/fi-fi/ammattilaiset/ai-audiology. Luettu: 8.4.2025

Sintonen, L., 2018, Keinotekoisten neuroverkkojen hyödyntäminen automaattisessa lintujen tunnistamisesssa äänen perusteella. Jyväskylän yliopisto. Saatavilla: https://jyx.jyu.fi/bitstream/handle/123456789/58825/URN%3ANBN%3Afi%3Ajyu-201807033457.pdf Luettu: 12.5.2025

TecnoDigital, 25.2.2024 Keinotekoiset hermoverkot: Kaikki mitä sinun tarvitsee tietää informatecdigital.com 2024. Hakupäivä 10.3.2025. https://informatecdigital.com/fi/keinotekoiset-neuroverkot-kaikki-mit%C3%A4-sinun-tarvitsee-tiet%C3%A4%C3%A4/ Luettu: 12.5.2025

Toft, R. (5.4.2024) Digital Audio: Sampling, Dither, Aliasing, and Bit Depth. Pdf-tiedosto. Western University. Luettavissa: https://ir.lib.uwo.ca/popmusicforum_techmatters/11 Luettu: 12.5.2025

Wildlife Acoustics Inc. (2025) What is an Audio Channel?, Wildlife Acoustics. Saatavilla: https://www.wildlifeacoustics.com/resources/faqs/what-is-an-audio-channel Luettu: 8.5.2025

Xiph.Org Foundation (2017) RNNoise. Saatavilla: https://github.com/xiph/rnnoise Luettu: 8.4.2025