# Aplikacija za čitanje vijesti realizirana modelom dubokog učenja

**Jurišić, Blaž**



**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**Sveučilišni studij**

# Aplikacija za čitanje vijesti realizirana modelom dubokog učenja

**Diplomski rad**

**Blaž Jurišić**

**Osijek, godina 2022.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

**Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit**

**Osijek, 16.09.2022.**

**Odboru za završne i diplomske ispite**

# Imenovanje Povjerenstva za diplomski ispit

| | |
|---|---|
| **Ime i prezime Pristupnika:** | Blaž Jurišić |
| **Studij, smjer:** | Diplomski sveučilišni studij Računarstvo |
| **Mat. br. Pristupnika, godina upisa:** | D-987R, 09.10.2018. |
| **OIB studenta:** | 92343532885 |
| **Mentor:** | Izv.prof.dr.sc. Ratko Grbić |
| **Sumentor:** | Matej Džijan, mag. ing. comp. |
| **Sumentor iz tvrtke:** | |
| **Predsjednik Povjerenstva:** | izv. prof. dr.sc. Josip Job |
| **Član Povjerenstva 1:** | Izv.prof.dr.sc. Ratko Grbić |
| **Član Povjerenstva 2:** | Izv. prof. dr. sc. Emmanuel-Karlo Nyarko |
| **Naslov diplomskog rada:** | Aplikacija za čitanje vijesti realizirana modelom dubokog učenja |
| **Znanstvena grana diplomskog rada:** | **Umjetna inteligencija (zn. polje računarstvo)** |
| **Zadatak diplomskog rada:** | Ovaj diplomski rad bavi se područjem sinteze govora iz teksta korištenjem dubokih neuronskih mreža. Cilj rada je izrada aplikacije koja olakšava čitanje vijesti aktualnih hrvatskih medija koje će se dohvaćati putem RSS izvora. Korisnik će moći odabrati željene medije s liste ponuđenih medija te preslušavati vijesti koje će izgovarati TTS (tekst u govor) model koji je realiziran korištenjem dubokih neuronskih mreža. Glas će biti realiziran za hrvatski jezik, baziran na podacima koji će se dohvaćati sa javno dostupnih izvora. Izgrađene modele i cjelokupnu aplikaciju potrebno je evaluirati na odgovarajući način. Sumentor: Matej Džijan Tema rezervirana za: Blaž Jurišić |
| **Prijedlog ocjene pismenog dijela ispita (diplomskog rada):** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 3 bod/bodaPostignuti rezultati u odnosu na složenost zadatka: 3 bod/bodaJasnoća pismenog izražavanja: 2 bod/bodaRazina samostalnosti: 3 razina |
| **Datum prijedloga ocjene od strane mentora:** | 16.09.2022. |

| | |
|---|---|
| Potvrda mentora o predaji konačne verzije rada: | *Mentor elektronički potpisao predaju konačne verzije.* |
| | Datum: 29.9.2022. |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 29.09.2022.**

| | |
|---|---|
| **Ime i prezime studenta:** | Blaž Jurišić |
| **Studij**: | Diplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | D-987R, 09.10.2018. |
| **Turnitin podudaranje [%]:** | 10 |

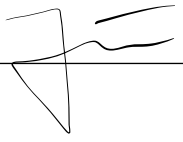
Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za čitanje vijesti realizirana modelom dubokog učenja**

izrađen pod vodstvom mentora Izv.prof.dr.sc. Ratko Grbić

i sumentora Matej Džijan, mag. ing. comp.

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija. Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:



# SADRŽAJ

# 1. UVOD

Vijest je informacija o nekom događaju, osobi ili pojavi. Može biti pisana, govorna ili vizualna. Radio vijest je sažetija i izraženija od pisane vijesti te korisniku omogućava veću slobodu pri obavljanju ostalih radnji pa je samim tim jednostavnija za konzumaciju od pisane. Suvremeni internetski portali vijesti dijele putem RSS toka pa ih je lagano integrirati u razna vizualna i zvučna sučelja. Kako bi se zvučno sučelje realiziralo, potrebno je koristiti takozvane metode prebacivanja teksta u govor (engl. *Text To Speech - TTS*) [1].

Tema ovog diplomskog rada je realizacija zvučnog sučelja iz tekstualnih RSS tokova putem tehnike dubokog učenja. Dubokim učenjem, model teksta u govor na skupu podataka govornika naučio je sintetizirati govor iz teksta dobivenim putem RSS tokova raznih hrvatskih portala za vijesti. Govor sintetiziran u aplikaciji simbolično podsjeća na govor poznatog hrvatskog TV voditelja Zorana Šprajca.

Motiv za pisanje diplomskog rada je sve veća popularnost zvučnih sučelja kako zbog lakšeg korištenja, tako i kao pomoć slijepim osobama koja se oslanjanju isključivo na takva sučelja. Modeli pretvaranja teksta u govor su tehnologija koja tek uzima zamah te tvrtke sve više upotrebljavaju iste u svojim proizvodima. Tako je i ovaj rad jedna od demonstracija takve tehnologije u jednoj od realnih primjena.

Naglasak diplomskog rada, u istraživačkom, tehničkom i sadržajnom smislu, stavljen je na rudarenje i obradu podataka te model dubokog učenja u trećem poglavlju, a evaluacija modela i kroz ankete opisana je u četvrtom poglavlju. U drugom poglavlju dan je pregled postojećih rješenja te osnovna terminologija koja olakšava daljnje čitanje rada. Fokus je veći na model prebacivanja teksta u govor u odnosu na aplikacijsko rješenje, zbog razlike u kompleksnosti između izrade modela dubokog učenja i izrade web aplikacije.

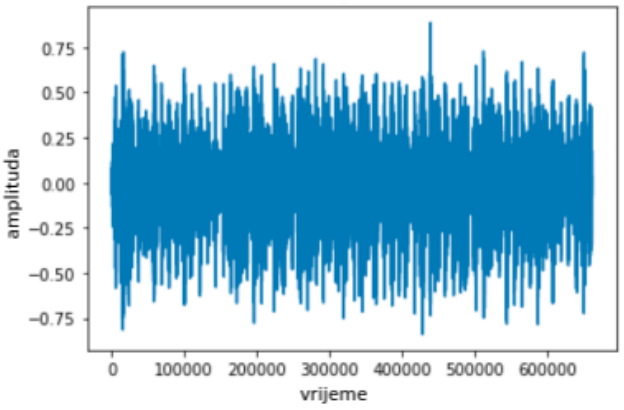
# 2. PREGLED POSTOJEĆIH RJEŠENJA ZA SINTEZU GOVORA I REALIZACIJU WEB APLIKACIJE

Ovo poglavlje sadržava osnovnu terminologiju koje olakšava daljnje čitanje, opisuje suvremena rješenja za sintezu govora te relevantne tehnologije za realizaciju web aplikacije. Također, opisana su rješenja generacije spektrograma iz teksta te modeli za sintezu govora iz spektrograma.

## 2.1. Osnovna terminologija

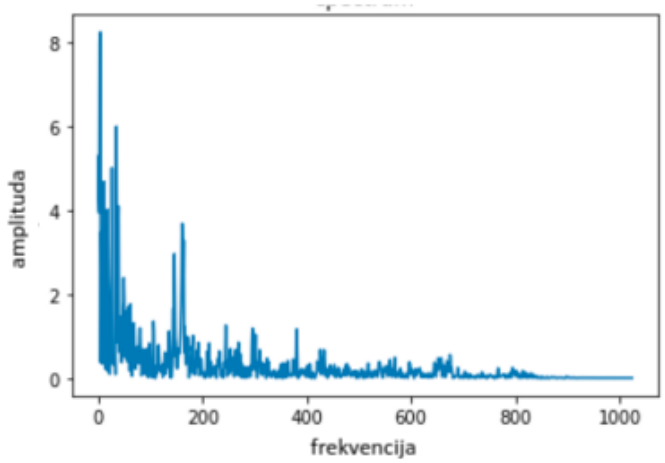
Suvremene duboke mreže za sintezu govora iz teksta uglavnom se baziraju na modelima realiziranim pomoću dubokih neuronskih mreža. Mreže koriste parove podataka govora i pripadajućeg transkripta teksta određenog govornika. Tijekom treninga, duboka mreža pokušava naučiti povezanost dijelova teksta sa dijelovima govora.

Zvuk kao podatak može biti prikazan u amplitudnom obliku kao na slici 2.1. na kojoj horizontalna os predstavlja vrijeme, a vertikalna os predstavlja amplitude. Također, zvuk kao podatak može biti prikazan u frekvencijskom obliku ukoliko se na njega primjeni Fourierova transformacija kao na slici 2.2.



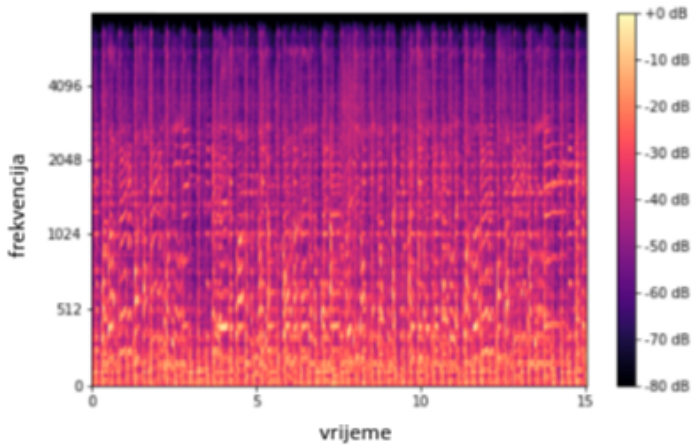

**Slika 2.1.** Prikaz zvučnog zapisa u amplitudnoj domeni

Na slici 2.2. horizontalna os predstavlja frekvencijski raspon dok vertikalna os prikazuje razinu amplitude.



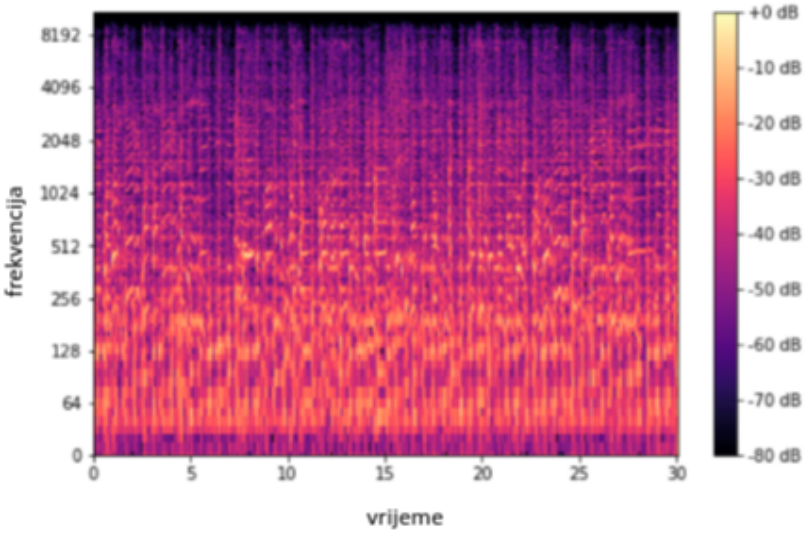

**Slika 2.2.** Prikaz zvučnog zapisa u frekvencijskoj domeni

Spektrogram zvuka, prikazan na slici 2.3. na svojoj horizontalnoj x osi sadrži vrijeme, na vertikalnoj y osi sadrži frekvenciju, a z os koja je realizirana spektrom boje, označava jačinu signala.




**Slika 2.3.** Prikaz govora spektrogramom

Optimalni spektrogram za prikaz govora prikazan na slici 2.4. je mel spektrogram, odnosno spektrogram prikazan u logaritamskoj skali iz razloga što ljudi zvuk ne percipiraju u linearnoj skali, odnosno percipiraju ga bliže logaritamskoj skali - zvukove niže frekvencije percipiraju puno jasnije od zvukova visokih frekvencija.




**Slika 2.4.** Prikaz govora mel spektrogramom

## 2.2. Pregled suvremenih arhitektura za sintezu govora

Suvremene arhitekture dubokih neuronskih mreža konstruirane su iz dva dijela: generator spektrograma te dekoder spektrograma, a za prikaz govora koriste spektrogram koji predstavlja frekvencijski oblik zvuka u kombinaciji s jačinom signala - amplitudom.

### 2.2.1. Generatori spektrograma

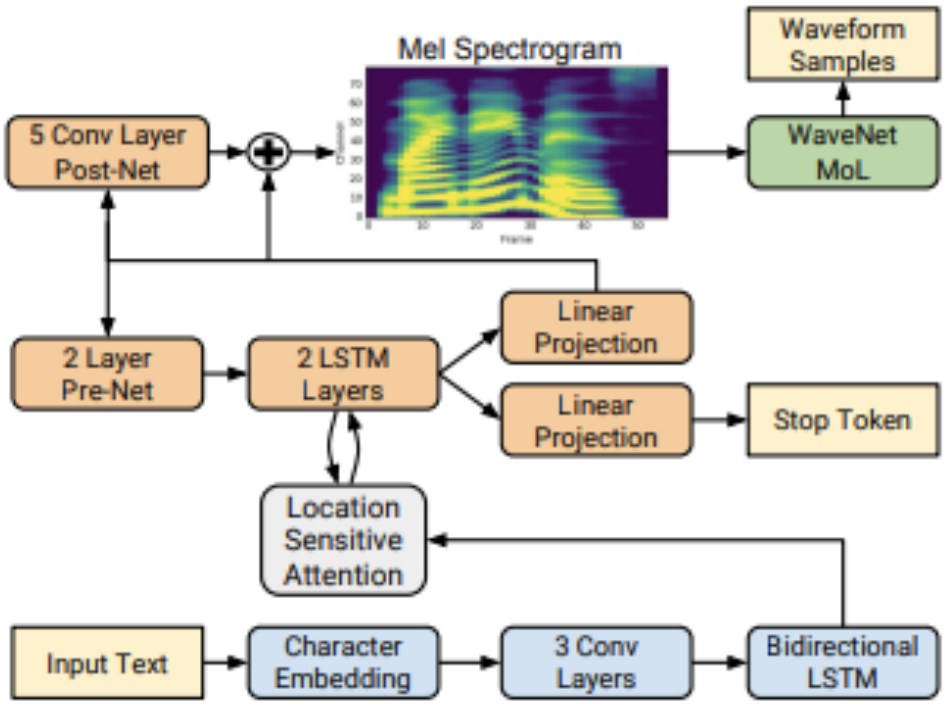
Generator spektrograma je mreža koja se uči na parovima transkripta govora i audio zapisima istog. Koristi koder - dekoder arhitekturu.

**Tacotron 2**

Tacotron 2 [3] mreža za generaciju spektrograma sastoji se od kodera i dekodera s modelom pažnje (engl. *attention network*) kao što je prikazano na slici 2.5. Koder pretvara niz znakova u skrivenu reprezentaciju koju dekoder koristi za predviđanje spektrograma. Ulazni znakovi uobličeni su pomoću naučenog **512-dimenzionalnog tenzora**, koji prolazi kroz stog od **tri konvolucijska sloja** od kojih svaki sadrži **512 filtera oblika 5 × 1** što znači da svaki filter obuhvaća 5 znakova, nakon čega slijedi **skupna normalizacija** i **ReLU aktivacija**. Konvolucijski slojevi modeliraju dugoročniji kontekst (N - grame) u ulaznom slijedu znakova. Izlaz konačnog konvolucijskog sloja prosljeđuje se u **jedan dvosmjerni LSTM** sloj koji sadrži **512 jedinica** (256 u svakom smjeru) za generiranje kodiranih značajki.

Izlaz kodera ulazi u mrežu pažnje (engl. *attention network*) koja sažima cijeli kodirani niz kao vektor konteksta fiksne duljine za svaki izlazni korak dekodera. Korištena je i pozornost osjetljiva na poziciju (engl. *location sensitive attention*) koja proširuje mehanizam pažnje tako da koristi kumulativne težine pažnje iz prethodnih vremenskih koraka dekodera kao dodatnu značajku. Vjerojatnosti pozornosti izračunavaju se nakon projiciranja ulaznih podataka i značajki pozicije na skrivene reprezentacije. Mjesta značajki se izračunavaju korištenjem 32 1-D konvolucijska filtera duljine 31.

Dekoder je autoregresivna ponavljajuća (engl. *recurrent*) neuronska mreža koja predviđa mel spektrogram iz kodiranog ulaznog vektora. Predviđeni rezultat iz prethodnog vremenskog koraka je prvo prošao kroz malu predmrežu (engl. *pre-net*) koja sadrži dva potpuno povezana sloja (engl. *fully connected layer*) od 256 skrivenih ReLU jedinica. Predmreža koja djeluje kao informacijsko usko grlo bitno je za učenje pažnje. Izlaz predmreže i vektor konteksta pozornosti su spojeni i proslijeđeni kroz stog od 2 jednosmjerna LSTM sloja s 1024 jedinice. LSTM izlaz i kontekst pažnje povezuju se u vektor se projicira kroz linearnu transformaciju za predviđanje finalnog okvira spektrograma. Konačno, predviđeni mel spektrogram prolazi kroz 5-slojnu konvolucijsku naknadnu mrežu (engl. *post-net*) koja predviđa preostali dio koji dodaje predviđanju za poboljšanje ukupne rekonstrukcije.

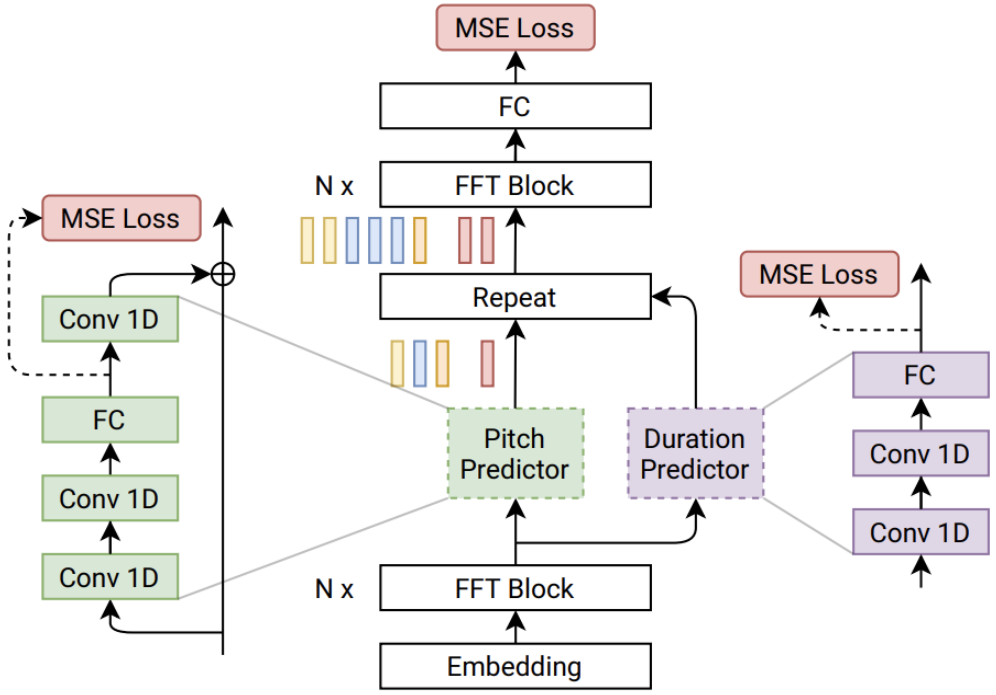



**Slika 2.5.** Prikaz arhitekture Tacotron 2 modela [3]

**FastPitch**

FastPitch [4] je potpuno paralelni model pretvaranja teksta u govor. Specifičan je po tome što tijekom treniranja uči konturu osnovne frekvencije govora što mu naposlijetku omogućava modifikaciju visine tona govora. Arhitektura FastPitch-a prikazana je na slici 2.6. FastPitch se sastoji od dva transformera. Prvi radi u domeni ulaznih tokena, drugi radi u domeni izlaznih okvira.

Prednosti FastPitch modela su brzina inferencije, paralelna obrada više ulaza odjednom te sposobnost modifikacije visine tona govora, a nedostatak taj što proizvodi govor koji nije prirodan kao primjerice govor proizveden Tacotron 2 modelom.



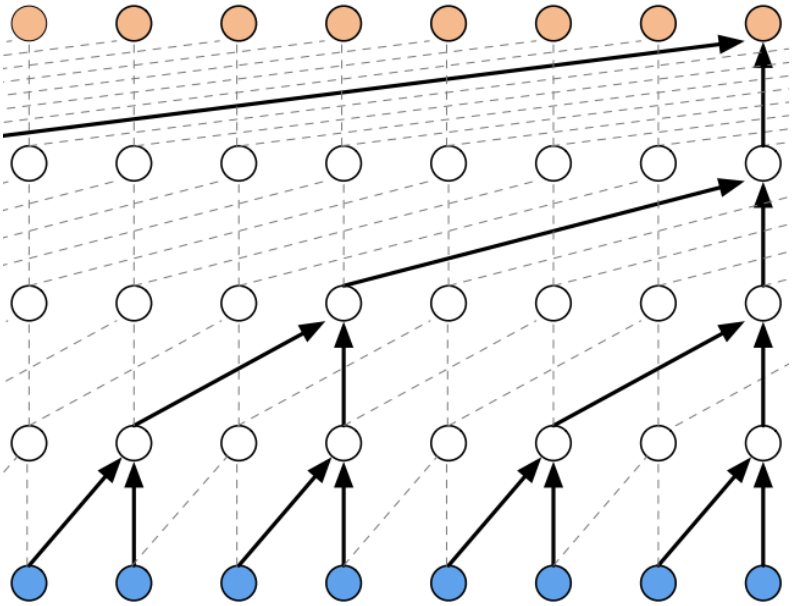

**Slika 2.6.** Prikaz arhitekture FastPitch modela [4]

### 2.2.2. Vokoderi

Dekoder ili vokoder (engl. *vocoder*) [2] je neuronska mreža neovisna o generatoru spektrograma koja radi inverznu Z transformaciju na danom spektrogramu - dani spektrogram sintetizira u zvučni zapis.

**WaveNet**

WaveNet [2] vokoder čini probabilistička i autoregresivna arhitektura. Cilj joj je generirati zvučne signale, odnosno audio zapise. Kao što je prikazano na slici 2.7, pri generiranju, svaki generirani korak ovisi o prijašnjem generiranom koraku. Plavi kružići predstavljaju ulazne uzorke, bijeli kružići predstavljaju skrivene konvolucijske slojeve, a narančasti kružići predstavljaju izlazne uzorke. Uvjetna vjerojatnost je modelirana u stog konvolucijskih slojeva.

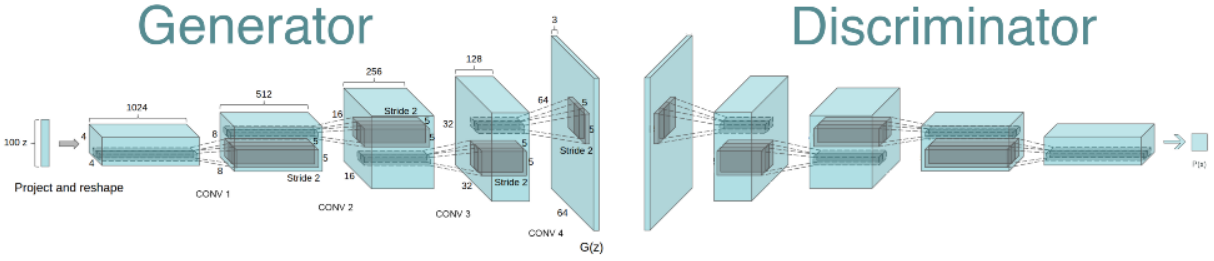


**Slika 2.7.** Prikaz arhitekture WaveNet modela [2]

Nedostatak WaveNeta je spor odziv koji je produkt autoregresivne arhitekture koju nije moguće optimizirati paralelizacijom, iz razloga što za svaki generirani okvir mreža uzima u obzir i prijašnji okvir.

**HiFiGAN**

HiFiGAN [5] je trenutno najbolja vokoderska arhitektura (SOTA) ukoliko se u obzir uzima omjer kvalitete proizvedenog zvuka sa brzinom generiranja zvuka iz spektrograma. Bazirana je na generativnoj suparničkoj mreži (engl. *Generative Adversarial Networks - GAN*), vrsti neuronskih mreža koja je posložena iz dvije sijamske mreže (engl. *Siamese Networks*) koje se međusobno nadmeću te na taj način treniraju jedna drugu. Jedna od dvije sijamske mreže zove se mreža generator, a druga mreža zove se diskriminator.




**Slika 2.8.** Prikaz arhitekture HiFiGAN modela [5]

U HiFiGAN arhitekturi prikazanoj na slici 2.8. generator kreira sintetički snimak iz spektrograma, a diskriminator pokušava predvidjeti radi li se o sintetičkoj ili stvarnoj snimci.

### 2.2.3. Cjeloviti modeli

**VITS**

VITS [6] arhitektura je napredna arhitektura koja je cjelovita u smislu da obuhvaća te trenira i generator i vokoder odjednom. VITS koristi HiFiGAN vokoder kao svoj dekoder. Prednost VITS modela je što nativno podržava više govornika, višejezičnost i kloniranje govora za govornike koji nisu uključeni u skup podataka za treniranje. U trenutku pisanja ovog diplomskog rada, VITS arhitektura poprilično je nova, izvorni kod nije objavljen te je teško reproducirati rezultate opisane u znanstvenom radu [6].

## 2.3. Postojeća rješenja za realizaciju web aplikacije

Web aplikacije danas vrlo su rasprostranjene i koriste se u širokoj upotrebi. Sukladno tome, postoji mnoštvo programskih okvira, alata i tehnologija za izradu istih. Web aplikacije uglavnom se izrađuju zadovoljavajući W3C standarde koje podržavaju svi poznatiji internetski preglednici. W3C standardi [7] definiraju stogove tehnologija od kojih je najpoznatiji HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets) i JS (JavaScript programski jezik).

Navedeni stogovi koriste se za izradu klijentskih aplikacija koje je izvode u internetskom pregledniku - prednji kraj (engl. *Front End*). Kako bi klijentska aplikacija mogla biti servirana krajnjem korisniku, ona mora imati i svoju podršku od strane serverske aplikacije - stražnji kraj (engl. *Back End*). Postoji mnoštvo programskih okvira za izradu web aplikacija koji automatiziraju i apstrahiraju proces izrade prednje i zadnje strane, a oni se uglavnom koriste u određenim domenama.

Streamlit [8] programski okvir dizajniran je za serviranje sučelja koje upravlja modelima strojnog učenja te modelima dubokog učenja. Sadrži gotove programske komponente koje se mogu upotrijebiti uz minimalnu količinu koda bez potrebe za pisanjem zasebnog klijentskog i serverskog koda.

# 3. PRIJEDLOG RJEŠENJA ZA SINTEZU GOVORA I REALIZACIJU WEB APLIKACIJE

pč

Poglavlje u svojem prvom pod poglavlju opisuje podatkovni skup, na koji je način skup prikupljen i označen te obrađen. Nadalje, opisan je proces treniranja mreže generatora spektrograma i vokodera za sintezu govora te su opisani provedeni eksperimenti sa različitim podatkovnim skupovima i različitim vrijednostima hiperparametara. U drugom potpoglavlju, opisana je realizacije web aplikacije.

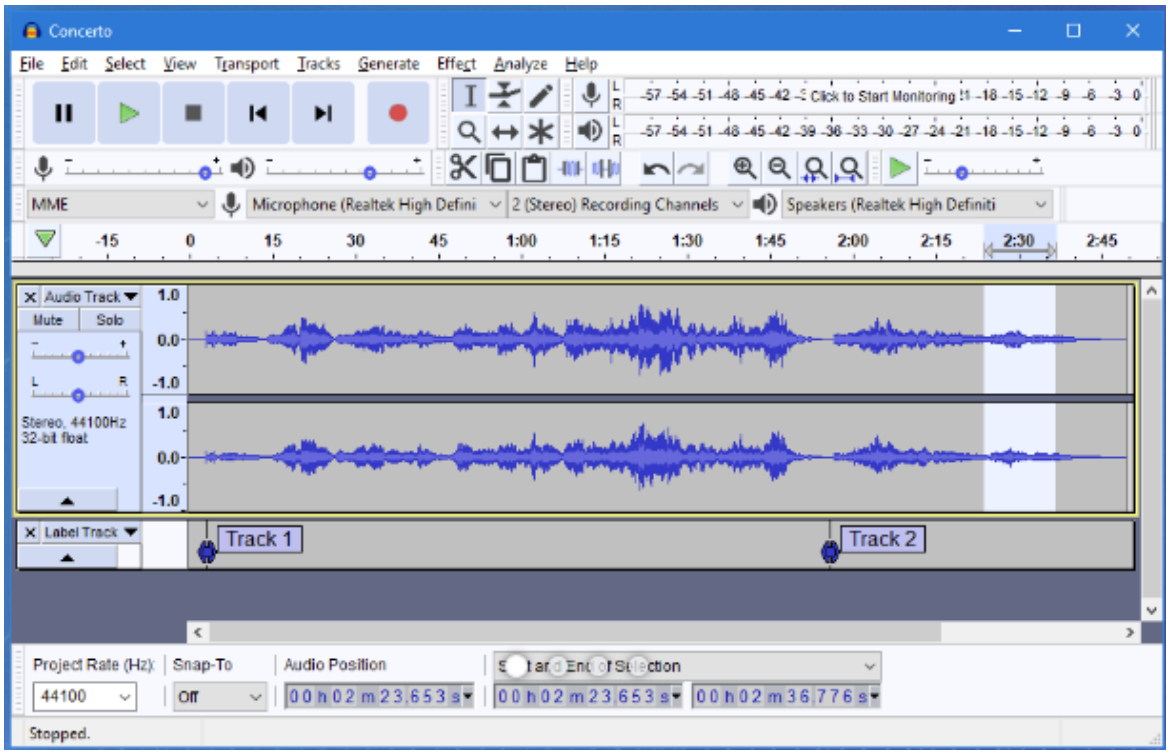
## 3.1. Prijedlog rješenja za sintezu govora

### 3.1.1. Izrada podatkovnog skupa za treniranje modela sinteze govora

Podatkovni skup prikupljen je na video platformi YouTube [9]. Prikupljeni su video zapisi emisija RTL Direkt, Stanje nacije, te intervju sa Zoranom Šprajcom u emisiji Podcast Inkubator.

**Prikupljanje podataka**

Video zapisi su preuzeti sa platforme YouTube, konvertirani iz .mp4 formata u .wav format te ručno izrezani u aplikaciji Audacity na način da su podijeljeni na duljine u trajanju od jedne do deset sekundi. Sučelje aplikacije Audacity prikazano je na slici 3.1.




**Slika 3.1.** Sučelje aplikacije Audacity [10]

Snimke su ručno pregledane i rezane na način da su ekstrahirani isključivo dijelovi govora, a snimke s pozadinskom bukom ili preklapanjem više govornika izbačene su iz skupa podataka.

### Označavanje podataka

Označavanje podataka obavljena je na način da se na cijeli skup audio snimki primjenio Google-ov model za prepoznavanje govora - ASR (engl. *automatic speech recognition*) [11]. Kod 3.1. sadržava programski kod koji poziva Google API za prepoznavanje govora.

**Kod 3.1.** Kod za komunikaciju sa Google Speech to text modelom

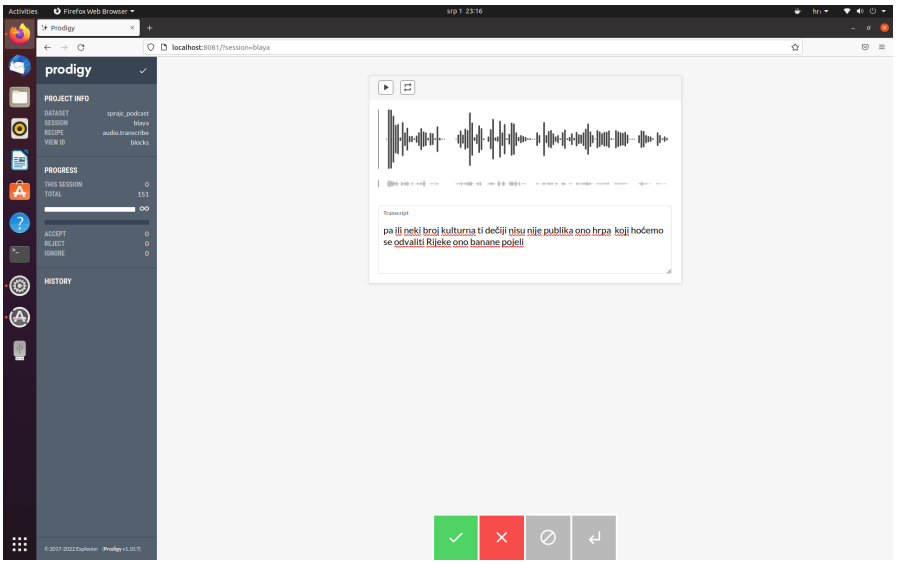
| ***Linija*** | ***Kod*** |
|---|---|
| 1: | `from google.cloud import speech` |
| 2: | `import os` |
| 3: | `import io` |

***Linija*** ***Kod***

```
4:    from tqdm import tqdm
5:    os.environ["GOOGLE_APPLICATION_CREDENTIALS"] =
      "<api-key>.json"
6:    client = speech.SpeechClient()
7:    config = speech.RecognitionConfig(
8:
      encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16
      ,
9:        audio_channel_count=1,
10:       language_code="hr-HR"  )
11:   filenames = !ls sn_mono_22050/
12:   file_names = list(map(lambda filename: "sn_mono_22050/"
      + filename, filenames))
13:   transcripts = []
14:   for file_name in tqdm(file_names):
15:       with io.open(file_name, "rb") as audio_file:
16:           content = audio_file.read()
17:           audio = speech.RecognitionAudio(content=content)
18:       response = client.recognize(request={"config":
      config, "audio": audio})
19:       for result in response.results:
20:
      transcripts.append(result.alternatives[0].transcript)
```

Takvi parovi audio zapisa i transkripta generiranih ASR modelom učitani su u Prodigy[12] sustav za označavanje podataka. Uz pomoć Prodigy sustava [12], čije je sučelje prikazano na slici 3.2, transkripti su ručno ispravljeni te je podatkovni skup eksportiran u .csv meta zapis koji

će naknadno biti korišten za vrijeme treniranja, a jednoznačno spaja audio zapis s transkriptom govornika. Inicijalni rezultati Google modela za prepoznavanje govora [11] uneseni su u platformu kao polazna točka. Transkripti dobiveni modelom naknadno su ispravljeni te je na taj način ubrzan i olakšan proces označavanja u kojoj označavač podataka ispravlja greške modela za prepoznavanje govora bez da mora pisati transkript samostalno, kao što je vidljivo na slici 3.3.




**Slika 3.2.** Prikaz Prodigy [12] programskog sučelja


**Slika 3.3.** Okno za preslušavanje audio zapisa i upisivanje pripadajućeg transkripta

**Obrada podataka**

Podaci su obrađeni na način da su uzorkovani na 22050 Hz, prebačeni u MONO zvuk s jednim kanalom te kodirani sa 16 bita. Korišten je konzolni alat sox [13] kao što je opisano u kodu 3.2..

**Kod 3.2.** Kod za paralelnu obradu audio zapisa

| *Linija* | *Kod* |
|---|---|
| 1: | `ls \| parallel sox {} -r 22050 ../audio_mono_22050/{} remix 1` |

Također, podaci su preimenovani na način da im je ime numerirano kako bi bilo lakše pratiti redoslijed izvođenja u daljnjem procesu, npr: audio_0001.wav. Bash skripta korištena je za ovakvo preimenovanje, opisana u kodu 3.3.

**Kod 3.3.** Kod za preimenovanje i numeriranje audio zapisa

| *Linija* | *Kod* |
|---|---|
| 1: | `a=1` |
| 2: | `for i in *.wav; do` |
| 3: | `    new=$(printf "audio_%04d.wav" "$a") #broj će se postaviti s četiri znamenke` |
| 4: | `    mv -i -- "$i" "$new"` |
| 5: | `    let a=a+1` |
| 6: | `done` |

Konačni podatkovni skup, nakon rezanja zapisa u sitnije dijelove te označavanja, sastoji se od 3270 zapisa koji ukupno traju 4 sata, 52 minute i 42 sekunde. Količina podataka svake emisije zasebno te ukupna količina iskazani su u tablici 3.1.

**Tablica 3.1.** Sastav podatkovnog skupa

| Izvor / emisija | broj konačnih zapisa | Vremensko trajanje |
|---|---:|---:|
| RTL direkt | 1137 | 01:44:51 |
| Stanje nacije | 1134 | 01:41:07 |
| Podcast inkubator | 999 | 01:26:44 |
| **Ukupno** | **3270** | **04:52:42** |

### 3.1.2. Cjevovod i arhitektura

Cjevovod se sastoji od predobrade podataka, modela za generiranje spektrograma iz teksta te modela čiji je zadatak pretvoriti spektrogram u zvučni zapis.

Predobrada podataka uključuje stvaranje spektrograma iz zvučnih zapisa te učitavanje istih u tenzore koji su potom pogodni za učitavanje na grafičku karticu, kao što je opisano u kodu 3.4. i u kodu 3.5. Također, pripadajući transkript se obrađuje, čisti i normalizira kao što je opisano u kodu 3.6. i 3.7. Kod za učitavanje podataka koristi biblioteku PyTorch [14].

**Kod 3.4.** kod za učitavanje parova teksta i govora

```
Linija  Kod
1:      def get_mel_text_pair( audiopath_and_text):
2:          audiopath, text = audiopath_and_text[0],
        audiopath_and_text[1]
3:          mel = self.get_mel(audiopath)
4:          text = self.get_text(text)
5:          return (text, mel)
```

**Kod 3.5.** Kod za pretvorbu audio zapisa u mel spektrogram

| *Linija* | *Kod* |
|---|---|
| 1: | `def get_mel(self, filename):` |
| 2: | `    audio, sampling_rate = load_wav_to_torch(filename)` |
| 3: | `    audio_norm = audio / self.max_wav_value` |
| 4: | `    audio_norm = audio_norm.unsqueeze(0)` |
| 5: | `    audio_norm = torch.autograd.Variable(audio_norm, requires_grad=False)` |
| 6: | `    melspec = self.stft.mel_spectrogram(audio_norm)` |
| 7: | `    melspec = torch.squeeze(melspec, 0)` |
| 8: | `    return melspec` |

**Kod 3.6.** Kod za pretvorbu teksta u tenzor

| *Linija* | *Kod* |
|---|---|
| 1: | `def get_text(self, text):` |
| 2: | `    text_norm = torch.IntTensor(text_to_sequence(text, self.text_cleaners))` |
| 3: | `    return text_norm` |

**Kod 3.7.** Kod za normalizaciju teksta

| *Linija* | *Kod* |
|---|---|
| 1: | `def croatian_cleaners(self, text):` |
| 2: | `    text = lowercase(text)` |
| 3: | `    text = collapse_whitespace(text)` |

| *Linija* | *Kod* |
|---|---|
| 4: | `return text` |

Funkcija za normalizaciju teksta čisti dodatne razmake i prazna polja te velika slova pretvara u mala. Grafemski zapis se dodatno ne pretvara u fonemski iz razloga što grafemski zapis hrvatskog jezika približno odgovara fonemskom zapisu (jedan glas je jedan znak). Izuzetak su digrami: LJ, NJ i DŽ. Model može zaključiti te specijalne slučajeve te nema potrebe za dodatnom fonemizacijom ili tokenizacijom digrama u posebne simbole.

Model za generiranje spektrograma iz teksta koristi Tacotron 2 [3] arhitekturu. Konkretno, koristi se račva Nvidia [15] implementacije Tacotron 2 [3] arhitekture dostupna na GitHub repozitoriju [16]. Korišteni model za pretvorbu spektrograma u zvučni zapis, odnosno vokoder, je račva HiFi-GAN modela [17].

### 3.1.3. Treniranje modela i odabir konačnog modela

Svi eksperimenti provedeni su sa podjelom podataka na skup podataka za treniranje i skup podataka za validaciju. Skup podataka za validaciju čini 5% ukupnih podataka, dok preostali udio predstavlja skup podataka za treniranje.

Funkcija gubitaka (engl. *loss function*) je funkcija koja računa razliku između stvarnog i generiranog podatka modela, a računa se kao negativna log-vjerojatnost stvarnog isječka. Treniranja svih modela i izvođenje svih opisanih eksperimenata izvode se na Ubuntu 20.0 Linux operacijskom sustavu te grafičkoj kartici Nvidia GeForce RTX 3080ti 12GB.

**Tacotron 2 generator mel spektrograma iz teksta**

Eksperimenti, u treniranju modela strojnog učenja, su iteracije treniranja koje su određene skupom podataka i skupom hiperparametra. Četiri eksperimenta provedena su pri treniranju modela generatora mel spektrograma iz teksta. Eksperimenti jedan, dva i tri pokretani su na podatkovnom skupu koji uključuje emisije “RTL direkt” [18] i “Stanje nacije” [19] zbog ujednačene prozodije. Četvrti eksperiment uključuje navedene emisije zajedno s intervjuom iz

emisije “Podcast Inkubator” [20]. Cilj ove podjele je kasnije usporediti dobivenu prozodiju. Za prvi trenirani skup podataka na modelu Tacotron 2 [3], provedena su tri eksperimenta.

**Prvi eksperiment**

Prvi eksperiment proveden je koristeći hiperparametre opisane u tablici 3.2.

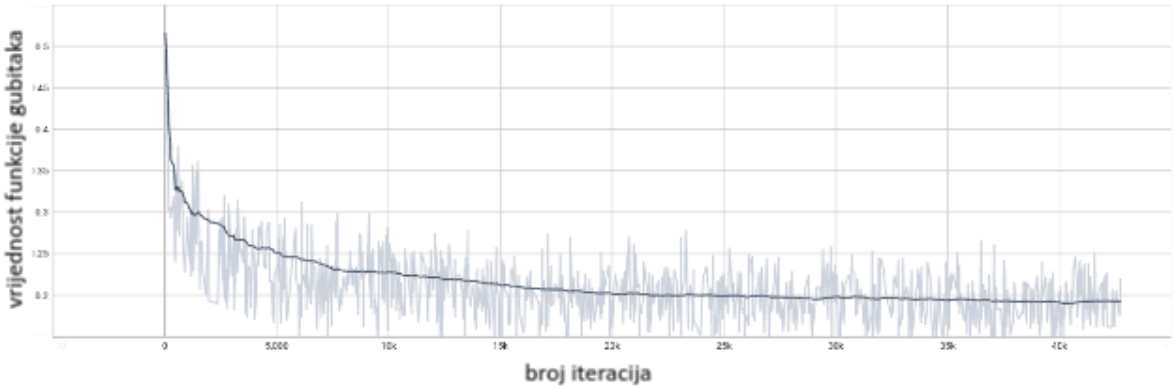
**Tablica 3.2.** hiperparametri prvog eksperimenta

| Hiperparametar | Vrijednost | Kratko objašnjenje |
|---|---|---|
| sampling_rate | 22050 | Broj uzorkovanja signala zvuka |
| win_length | 1024 | Širina prozora |
| hop_length | 256 | širina skoka prozora |
| n_mel_channels | 80 | Broj kanala mel spektrograma |
| mel_fmin | 0.0 | Minimalna frekvencija zvuka |
| mel_fmax | 8000.0 | Maksimalna frekvencija zvuka |
| symbols_embedding_dim | 512 | Dimenzija tenzora simbola |
| encoder_kernel_size | 5 | Dimenzija kernela kodera |
| encoder_n_convolutons | 3 | Broj konvolucijskih slojeva kodera |
| encoder_embedding_dim | 512 | Dimenzija tenzora kodera |
| decoder_rnn_dim | 1024 | LSTM čvorovi u koderu |
| prenet_dim | 256 | ReLU čvorovi u pre-net mreži |
| max_decoder_steps | 1000 | Broj koraka nakon kojeg će dekoder stati s dekodiranjem |
| gate_treshold | 0.5 | Vrijednost koja kontrolira izvršenje generacije spektrograma |
| p_attention_dropout | 0.1 | Dropout mreže pažnje |
| p_decoder_dropout | 0.1 | Dropout dekodera |
| postnet_embedding_dim | 512 | Dimenzija tenzora post-net mreže |

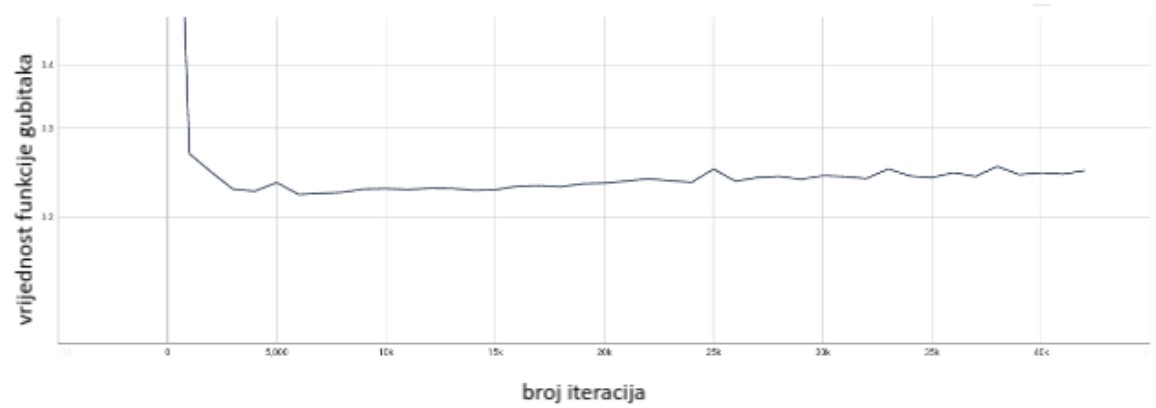
| | | |
|---|---|---|
| postnet_kernel_size | 5 | Dimenzija kernela konvolucije post-net mreže |
| postnet_n_convolutions | 5 | Broj konvolucijskih slojeva post-net mreže |
| learning_rate | 1e-3 | Brzina treniranja, veličina koraka pri traženju minimuma funkcije gubitaka |
| batch_size | 8 | Broj ulaznih tenzora u jednoj iteraciji |
| epochs | 500 | Broj epoha nakon kojeg će model sam stati s treniranjem |
| iters_per_checkpoint | | Broj iteracija nakon kojeg se vrši proces validacije |
| distributed_run | False | Koristi li model više grafičkih kartica |

Pri pokretanju, model je treniran metodom prijenosa znanja u kojem su se koristile težine modela treniranog na LJSpeech [21] podatkovnom skupu. Podatkovni skup LJSpeech je korpus engleske govornice [21] koji je sadržan od trinaest tisuća i sto zapisa u trajanju od jedne do deset sekundi. Ukupno vremensko trajanje tih podataka je oko dvadeset i četiri sata. Ignorirane su težine govornika pa razlike u jeziku i spolu ne stvaraju poteškoće u prijenosu znanja.

Proces treninga je zaustavljen nakon četrdeset dvije tisuće iteracija nakon što se trenirao gotovo dvadeset jedan sat. Model je uspješno konvergirao, a svoj minimum, funkcija gubitaka na skupu podataka za trening, dosegla je nakon šest tisuća iteracija, kao što je vidljivo na slikama 3.4. i 3.5.



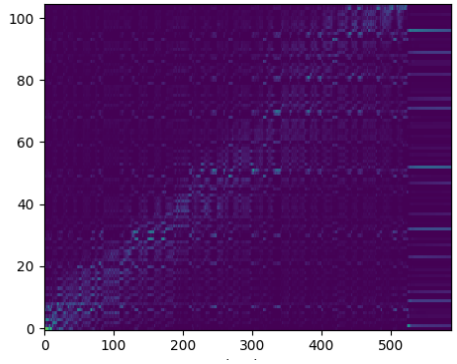
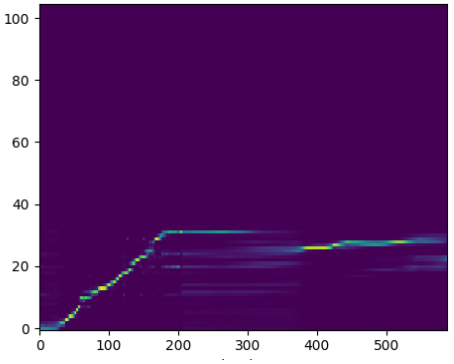
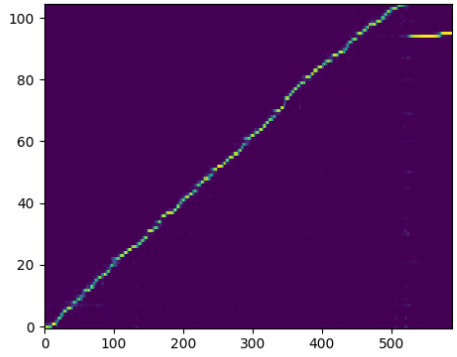
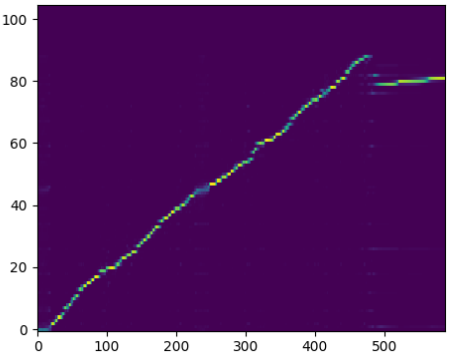

**Slika 3.4.** Vrijednost funkcije gubitaka na trening skupu za eksperiment 1

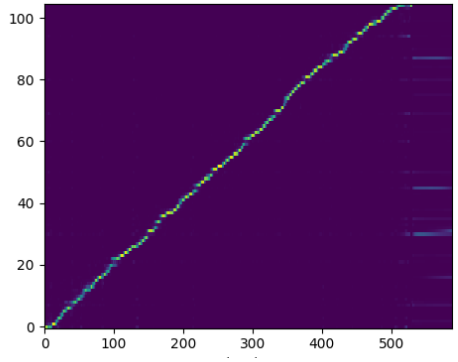
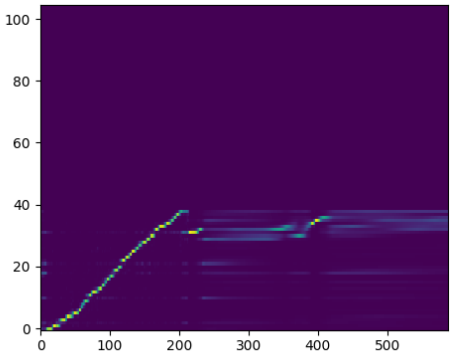
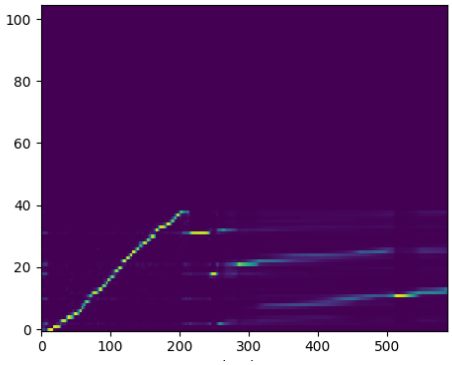
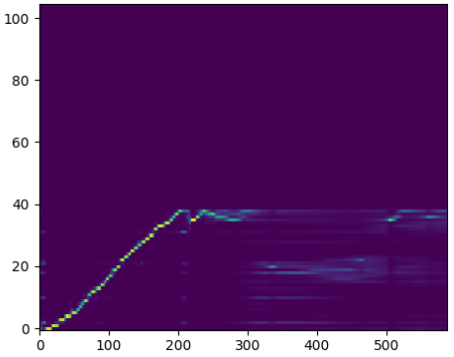



**Slika 3.5.** Vrijednost funkcije gubitaka na validacijskom skupu za eksperiment 1

Graf poravnanja, tablica 3.2, je graf koji prikazuje slijed težina pažnji iz dekodera. On je pokazatelj koliko su okviri predviđenog spektrograma povezani s prijašnjim predviđenim okvirima. Idealan graf poravnanja je sporedna dijagonala matrice. Što je graf sličniji sporednoj dijagonali, to je zvuk u pravilu bolje generiran. Vertikalna os na grafu poravnanja prikazuje korak kodera, a horizontalna os korak dekodera. Grafovi poravnanja računati su za rečenice iz validacijskog skupa.

**Tablica 3.2.** Grafovi ponavljanja za Eksperiment 1

| **Eksperiment 1** | |
|---|---|
|  |  |
| 1000 iteracija | 14000 iteracija |
|  |  |
| 16000 iteracija | 19000 iteracija |

| | |
|---|---|
|  |  |
| 20000 iteracija | 25000 iteracija |
|  |  |
| 27000 iteracija | 33000 iteracija |

Iz rezultata je jasno vidljivo da se poravnanje gubi nakon dvadeset tisuća iteracija te model nakon te točke postaje lošiji.

**Drugi eksperiment**

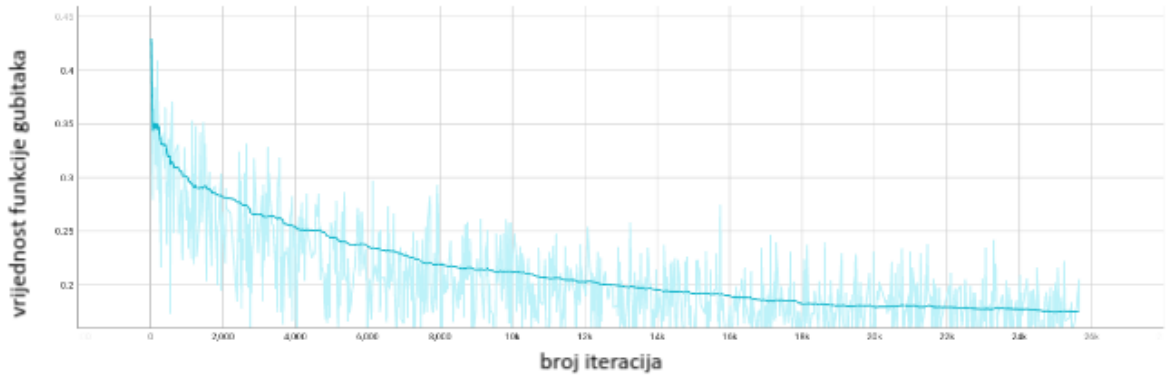
Drugi eksperiment razlikuje se od prvog po hiperparametrima stope učenja (engl. *learning rate*), napuštanja poravnanja (engl. *attention dropout*) i napuštanja dekodera (engl. *decoder dropout*).

U znanstvenom radu [22] opisana je metoda idealne vrijednosti stope učenja za mrežu pažnje u odnosu na hiperparametar veličine hrpe (engl. *batch size*). Funkcija je opisana jednadžbom 3.1.

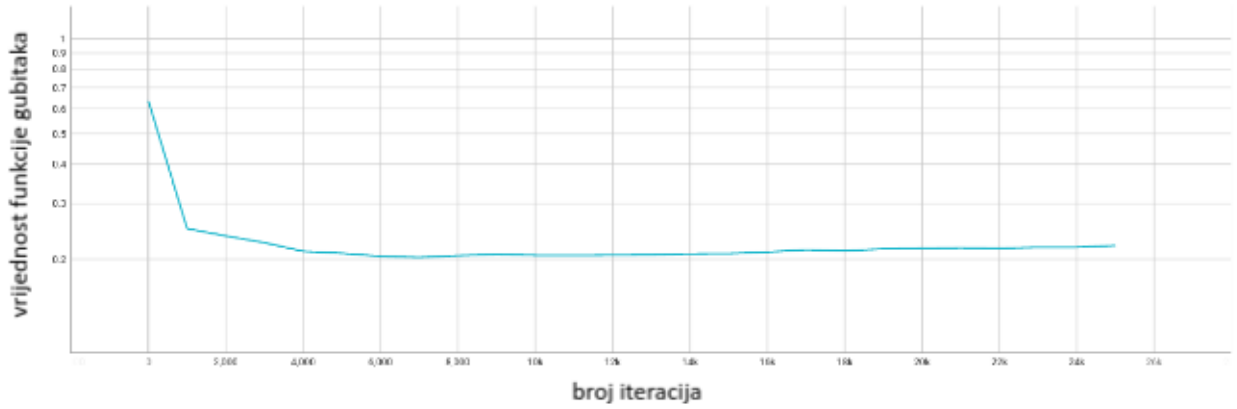
$$stopa\ učenja\ =\ 0.001\ \cdot\ \sqrt{(\frac{veličina\ hrpe}{256})} \tag{3-1}$$

Hiperparametar veličine hrpe iznosi 8, a on ovisi o grafičkoj kartici korištenoj u eksperimentu, pa analogno formuli, vrijednost hiperparametra stope učenja iznosi 1.77e-4. Hiperparametri napuštanja poravnanja i napuštanja dekodera promjenjeni su sa 0.1 na 0.3.

Treniranje drugog eksperimenta zaustavljeno je nakon 25000 iteracija jer je prijašnjim eksperimentom utvrđeno da model na podacima počinje degradirati nakon te brojke. Treniranje je trajalo točno trinaest sati, a grafovi funkcije gubitaka za trening i validaciju vidljivi su na slikama 3.6. i 3.7. Grafovi Poravnanja prikazani su u tablici 3.3.

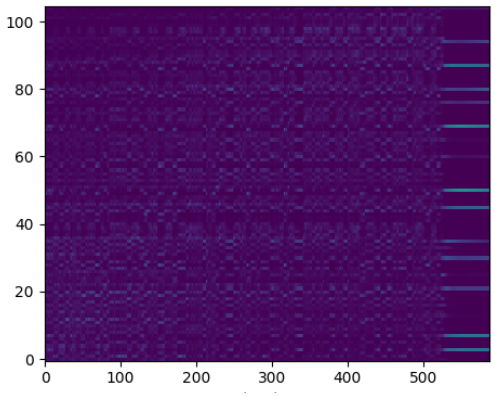
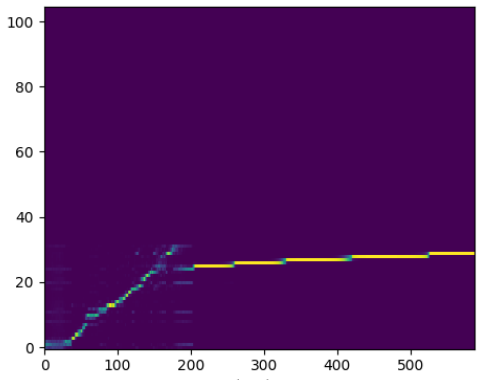


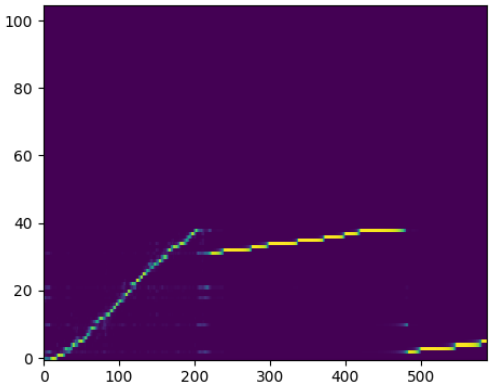
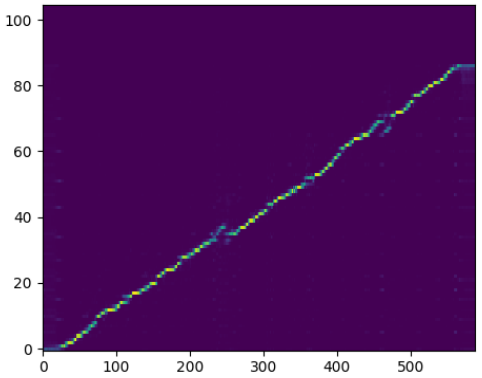
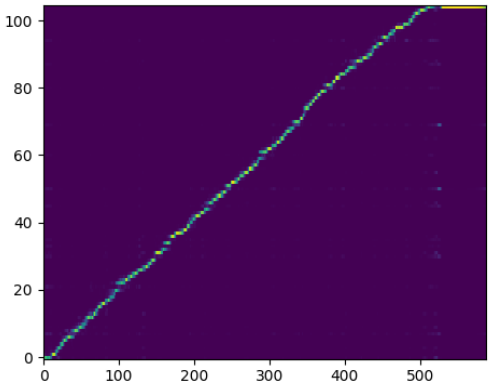
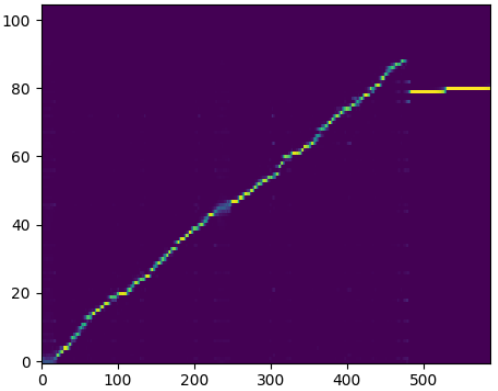
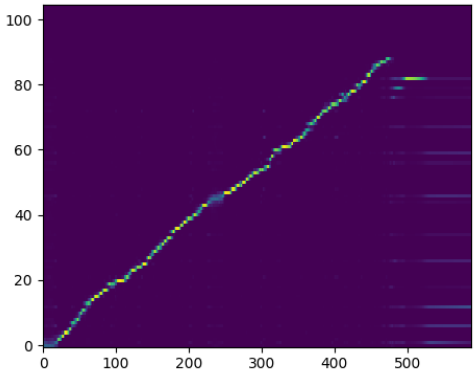
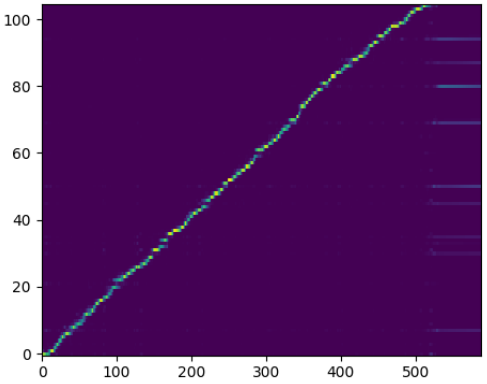

**Slika 3.6.** Vrijednost funkcije gubitaka na trening skupu za eksperiment 2




**Slika 3.7.** Vrijednost funkcije gubitaka na validacijskom skupu za eksperiment 2

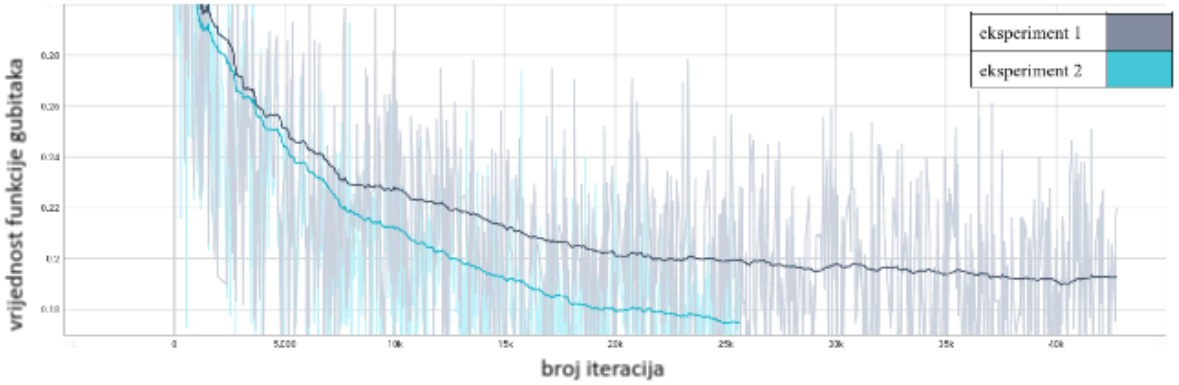
**Tablica 3.3.** Grafovi ponavljanja za Eksperiment 2

| **Eksperiment 2** | |
|---|---|
|  |  |
| 1000 iteracija | 6000 iteracija |

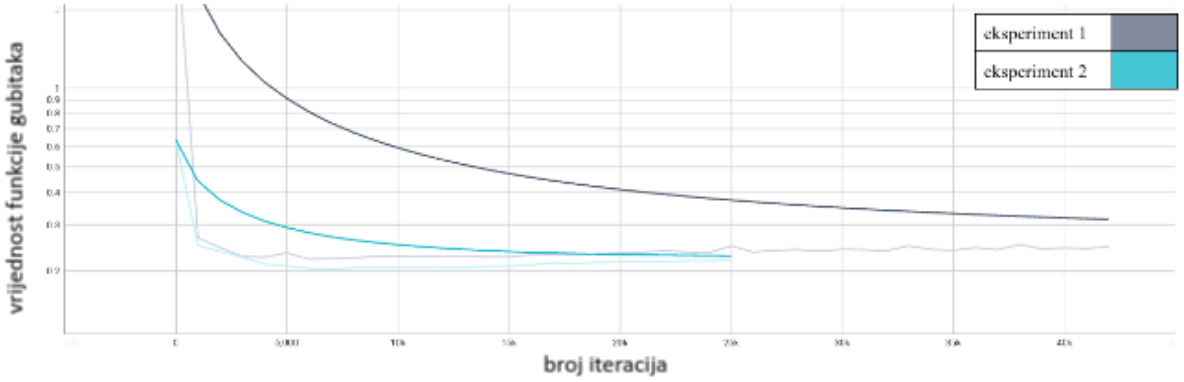
| | |
|---|---|
|  |  |
| 7000 iteracija | 15000 iteracija |
|  |  |
| 16000 iteracija | 19000 iteracija |
|  |  |

| 21000 iteracija | 25000 iteracija |
|---|---|

Drugi eksperiment konvergirao je brže nego prvi. Funkcija gubitaka daje manje rezultate tijekom treninga i validacije iz čega se može zaključiti da je promjena hiperparametara značajno doprinijela kvaliteti modela. Na usporednim grafovima funkcije gubitaka tijekom treniranja i validacije za oba grafa, na slikama 3.8. i 3.9, tamno plava krivulja prikazuje funkcije gubitaka za prvi eksperiment, a svijetlo plava krivulja prikazuje funkcije gubitaka za drugi eksperiment.




**Slika 3.8.** Vrijednost funkcije gubitaka na trening skupu za eksperiment 1 i 2

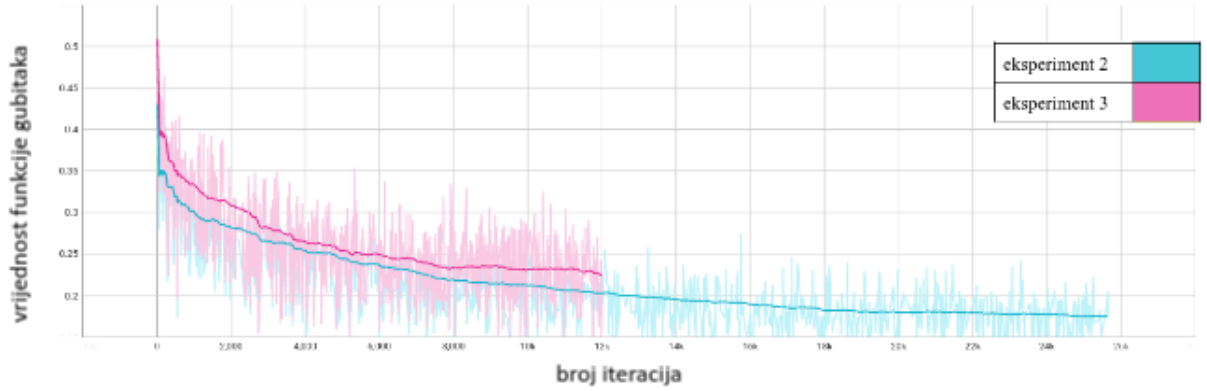



**Slika 3.9.** Vrijednost funkcije gubitaka na validacijskom skupu za eksperiment 1 i 2

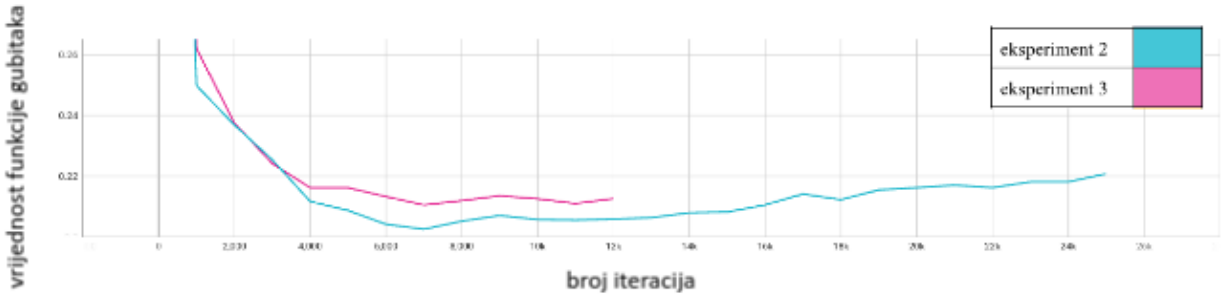
**Treći eksperiment**

Treći eksperiment koristi hiperparametre napuštanja poravnanja i napuštanja dekodera u iznosu od 0.1 kako bi se mogao izvesti zaključak poboljšavaju li dva navedena hiperparametra rezultat treniranog modela, ili je za to zaslužna isključivo promjena stope učenja, čija vrijednost ostaje 1.77e-4.

Treći eksperiment zaustavljen je nakon dvanaest tisuća iteracija jer je polučio slabije rezultate od drugog eksperimenta. Krivulja gubitaka, označena rozom krivuljom na slikama 3.10 i 3.11, značajno je lošija u trećem eksperimentu u odnosu na drugi eksperiment označen svijetlo plavom bojom na slikama 3.10. i 3.11, dok je krivulja gubitaka za validaciju neznatno lošija u trećem eksperimentu u odnosu na drugi.



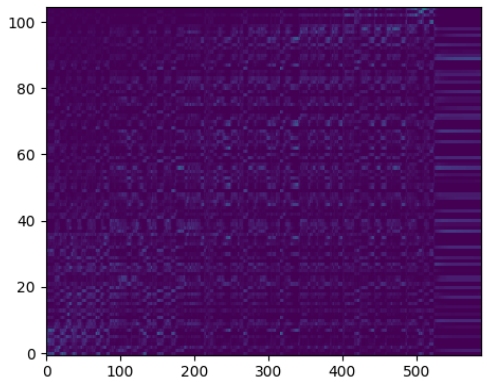
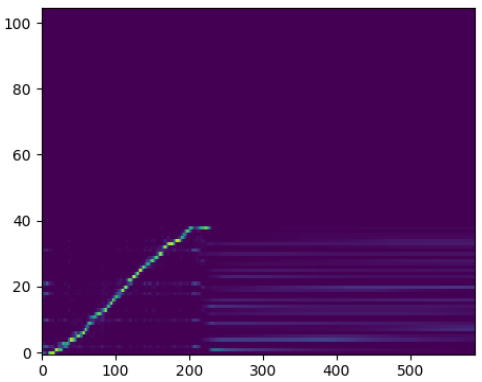

**Slika 3.10.** Vrijednost funkcije gubitaka na trening skupu za eksperiment 2 i 3

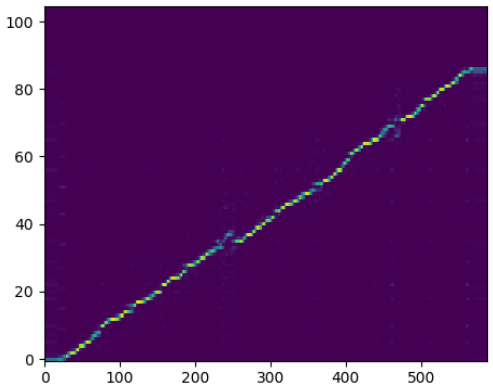
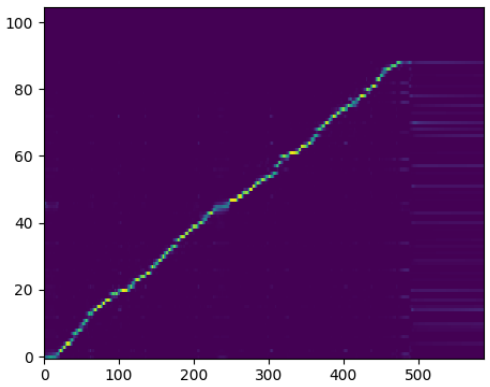



**Slika 3.11.** Vrijednost funkcije gubitaka na validacijskom skupu za eksperiment 2 i 3

Grafovi poravnanja za treći eksperiment, prikazani u tablici 3.4, izgledaju bolje nego u prethodna dva eksperimenta za isti broj iteracije. Jedan od mogućih razloga za to je da se rečenica u validacijskom postupku generira s određenim nasumičnim sjemenom (engl. *seed*), pa je generacija nedeterministička. Zbog toga, graf poravnanja nije dobar faktor pri uspoređivanju modela u ranim fazama treniranja dok model nije konvergirao, ali može biti dobar pokazatelj pretjeranog usklađivanja na podatke (engl. *overfitting*) u kasnijim fazama treniranja, nakon što je model konvergirao.

**Tablica 3.4.** Grafovi ponavljanja za Eksperiment 3

| **Eksperiment 3** | |
|---|---|
|  |  |

| 1000 iteracija | 4000 iteracija |
|---|---|
|  |  |
| 7000 iteracija | 12000 iteracija |

### Četvrti eksperiment

Četvrti eksperiment podrazumijeva učenje modela na skupu podataka svih emisija. Za četvrti eksperiment korišteni su hiperparametri prikazani u tablici 3.5.

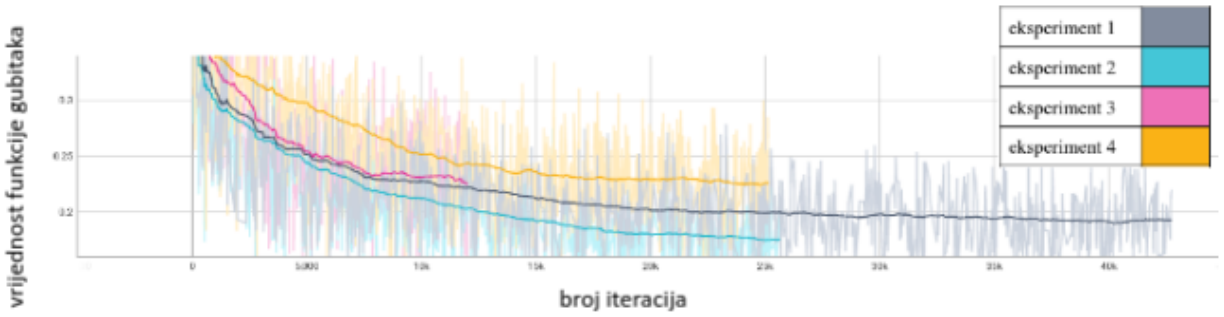
**Tablica 3.5.** Hiperparametri za eksperiment 4

| Hiperparametar | Vrijednost | Kratko objašnjenje |
|---|---|---|
| sampling_rate | 22050 | Broj uzorkovanja signala zvuka |
| win_length | 1024 | Širina prozora |
| hop_length | 256 | širina skoka prozora |
| n_mel_channels | 80 | Broj kanala mel spektrograma |
| mel_fmin | 0.0 | Minimalna frekvencija zvuka |
| mel_fmax | 8000.0 | Maksimalna frekvencija zvuka |

| | | |
|---|---|---|
| symbols_embedding_dim | 512 | Dimenzija tenzora simbola |
| encoder_kernel_size | 5 | Dimenzija kernela kodera |
| encoder_n_convolutons | 3 | Broj konvolucijskih slojeva kodera |
| encoder_embedding_dim | 512 | Dimenzija tenzora kodera |
| decoder_rnn_dim | 1024 | LSTM čvorovi u koderu |
| prenet_dim | 256 | ReLU čvorovi u pre-net mreži |
| max_decoder_steps | 1000 | Broj koraka nakon kojeg će dekoder stati s dekodiranjem |
| gate_treshold | 0.5 | Vrijednost koja kontrolira izvršenje generacije spektrograma |
| p_attention_dropout | 0.3 | Dropout mreže pažnje |
| p_decoder_dropout | 0.3 | Dropout dekodera |
| postnet_embedding_dim | 512 | Dimenzija tenzora post-net mreže |
| postnet_kernel_size | 5 | Dimenzija kernela konvolucije post-net mreže |
| postnet_n_convolutions | 5 | Broj konvolucijskih slojeva post-net mreže |
| learning_rate | 1.77e-4 | Brzina treniranja, veličina koraka pri traženju minimuma funkcije gubitaka |
| batch_size | 8 | Broj ulaznih tenzora u jednoj iteraciji |
| epochs | 500 | Broj epoha nakon kojeg će model sam stati s treniranjem |
| iters_per_checkpoint | 1000 | Broj iteracija nakon kojeg se vrši proces validacije |
| distributed_run | False | Koristi li model više grafičkih kartica |

Eksperiment je zaustavljen nakon 25 tisuća iteracija. Na slikama 3.12. i 3.13. vidimo da je model eksperimenta 4 najsporije konvergirao što je i očekivano s obzirom na uvedeni diverzitet u

podacima. Grafovi poravnanja za eksperiment 4 prikazani su u tablici 3.6. Model u kasnijoj konvergenciji ima vrlo slične gubitke kao i model eksperimenta 2.

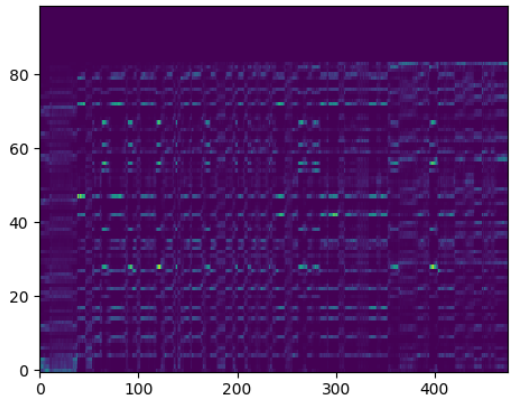



**Slika 3.12.** Vrijednost funkcije gubitaka na trening skupu za sva četiri eksperimenta

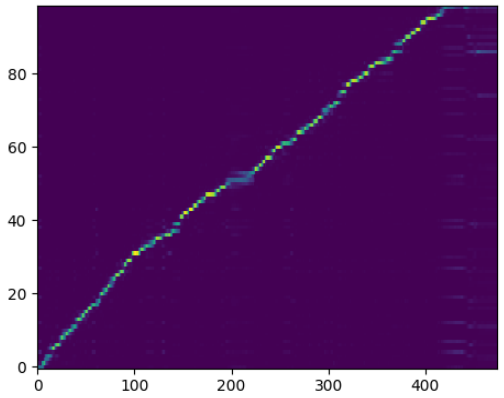

**Slika 3.13.** Vrijednost funkcije gubitaka na validacijskom skupu za sva četiri eksperimenta

**Tablica 3.6.** Grafovi ponavljanja za Eksperiment 4

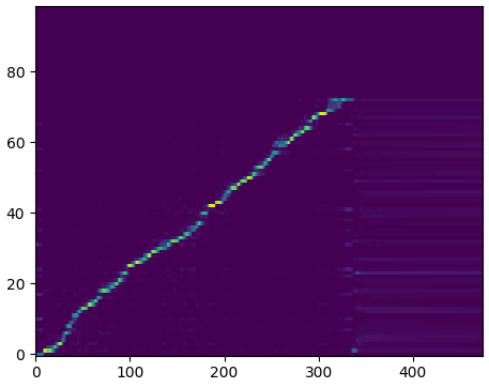
| **Eksperiment 4** |
|---|



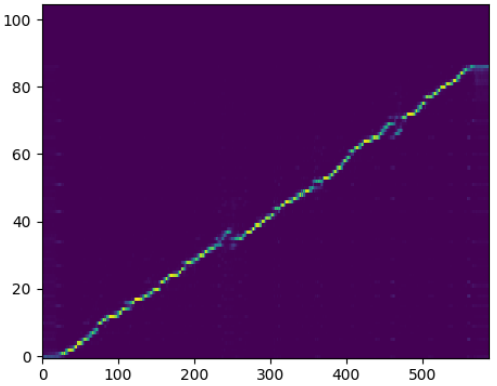

1000 iteracija



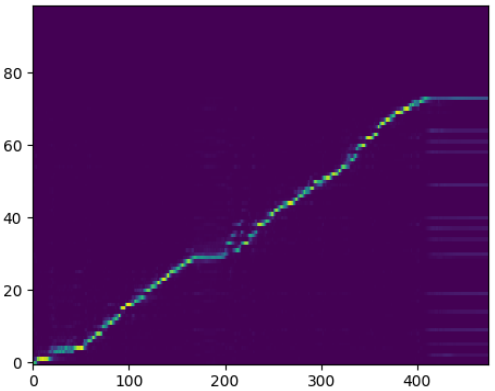

6000 iteracija

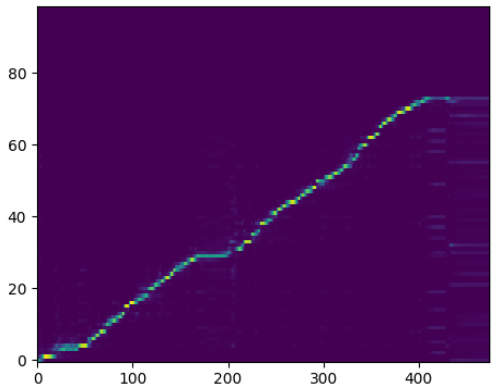


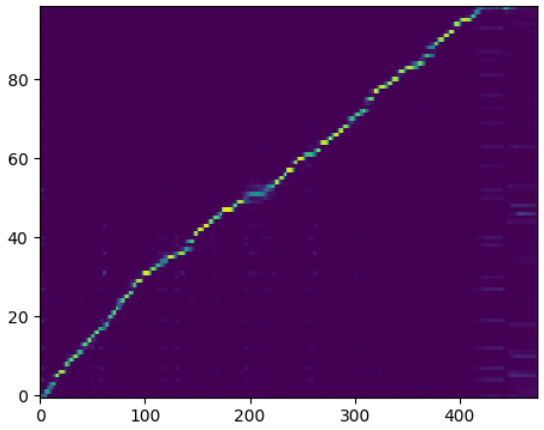
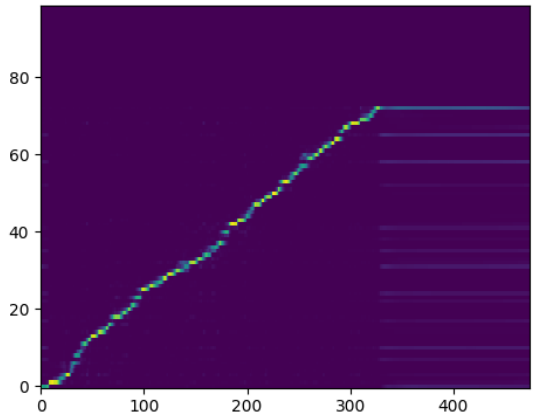

7000 iteracija




10000 iteracija



| 14000 iteracija | 16000 iteracija |
|---|---|
|  |  |
| 19000 iteracija | 25000 iteracija |

**HiFiGAN vokoder**

HiFiGAN vokoder [17] treniran je na svim prikupljenim podacima. Za treniranje vokodera potrebni su samo audio zapisi bez lingvističkih karakteristika, odnosno transkripta govora. Vokoder pretvara mel spektrogram kreiran Tacotron 2 [3] modelom u audio zapis wave formata.

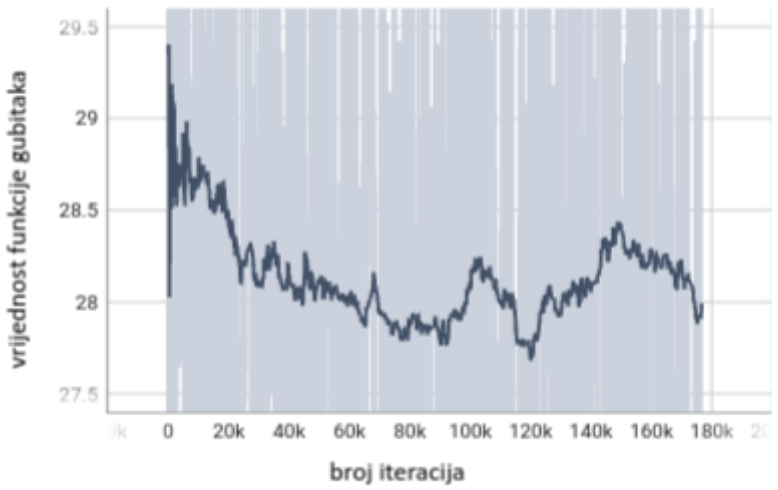
Model je treniran trinaest sati i pedeset dvije minute, u sto sedamdeset šest tisuća iteracija. Korišteni hiperparametri nalaze se u tablici 3.7. Treniranje se također kao i kod Tacotron 2, obavljalo tehnikom prijenosa znanja u kojem su se koristile težine modela treniranog na LJSpeech podatkovnom skupu [21].

**Tablica 3.7.** Hiperparametri za treniranje HiFiGAN vokodera [17]

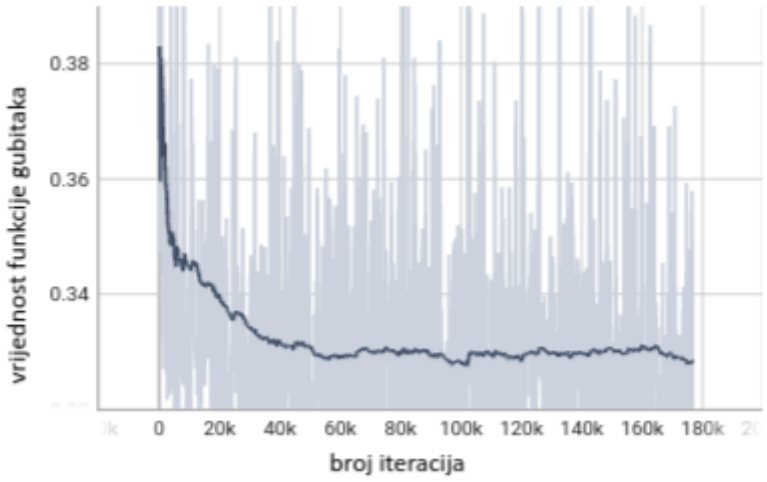
| Hiperparametar | Vrijednost | Kratko objašnjenje |
|---|---|---|
| sampling_rate | 22050 | Broj uzorkovanja signala zvuka |
| win_size | 1024 | Širina prozora |

| | | |
|---|---|---|
| hop_size | 256 | širina skoka prozora |
| n_mel_channels | 80 | Broj kanala mel spektrograma |
| fmin | 0.0 | Minimalna frekvencija zvuka |
| fmax | 8000.0 | Maksimalna frekvencija zvuka |
| n_fft | 1024 | Broj izlaznih frekvencijskih kutija |
| num_gpus | 1 | Broj grafičkih kartica |
| adam_b1 | 0.8 | adam optimizator osnovnog pravca 1 (engl. *baseline 1*) |
| adam_b2 | 0.88 | adam optimizator osnovnog pravca 2 (engl. *baseline 2*) |
| learning_rate | 3e-5 | Brzina treniranja, veličina koraka pri traženju minimuma funkcije gubitaka |
| batch_size | 4 | Broj ulaznih tenzora u jednoj iteraciji |

Grafovi funkcija gubitaka tokom treniranja prikazane su na slikama 3.14. i 3.15.




**Slika 3.14.** Vrijednost funkcije gubitaka na trening skupu za HiFiGAN model




**Slika 3.15.** Vrijednost funkcije gubitaka na validacijskom skupu za HiFiGAN model

Iz grafova gubitaka, vidljivo je da je već 80000 iteracija i više nego dovoljno za podešavanje modela, odnosno 6 sati treniranja na korištenom sklopovlju.

## 3.2. Prijedlog rješenja za web aplikaciju

Web aplikacija realizirana je u programskom okviru Streamlit [8]. Dohvaćani su RSS tokovi sa net.hr [23], index.hr [24] te 24 sata [25] portala u obliku XML fajlova. Zbog Streamlit arhitekture [8], potrebno ih je realizirati kao singleton funkcije koje se pozivaju isključivo jednom kako bi se moglo zapamtiti stanje trenutne vijesti za svaki portal.

**Kod 3.8.** Kod za singleton funkciju koja vraća vijesti iz RSS toka

| ***Linija*** | ***Kod*** |
|---|---|
| 1: | `@st.experimental_singleton` |
| 2: | `def read_rss_indexhr():` |
| 3: | `    feed = feedparser.parse("https://www.index.hr/rss/")` |
| 4: | `    feed_entries = feed.entries` |

```
Linija  Kod
5:          news = list(map(lambda entry:
        entry.title,feed_entries))
6:          return news
```

Nakon definiranja funkcije koja vraća singleton objekt, potrebno je isti inicijalizirati.

**Kod 3.9.** Inicijalizacija singleton objekta

```
Linija  Kod
1:      news_index = read_rss_indexhr()
```

Sučelje je jednostavan gumb koji se definira kao gotova Streamlit komponenta kojoj se predaje funkcija za izvršenje na klik "generate_index".

**Kod 3.10.** Kod za gumb na sučelju

```
Linija  Kod
1:      st.button("Pročitaj index.hr", on_click=generate_index)
```

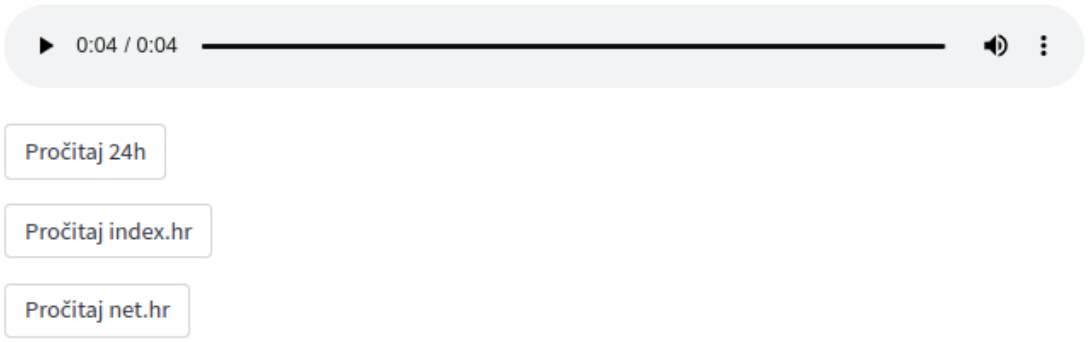
**Kod 3.11.** Kod za funkciju za izvršenje na klik

```
Linija  Kod
1:      def generate_index():
2:          audio, sr = end_to_end_infer(news_index.pop(),
        False, False,0)
3:          scipy.io.wavfile.write('tmp/index.wav', sr, audio)
```

***Linija*** ***Kod***

```
4:          st.audio('./tmp/index.wav')
```

Završno, funkcija za izvršenje na klik poziva internu funkciju za inferenciju istreniranih Tacotron 2 i HiFiGAN modela kojoj se predaje vijest dohvaćena iz RSS kanala u tekstualnom obliku, funkcija za inferenciju vraća audio signal koji se potom sprema kao wave datoteka u priveremeni direktorij, a Streamlit audio komponenta čita datoteku i prikazuje audio komponentu u sučelju. Izgled grafičkog sučelja prikazan je na slici 3.16.




**Slika 3.16.** Prikaz grafičkog sučelja web aplikacije

# 4. EVALUACIJA PREDLOŽENOG RJEŠENJA ZA SINTEZU GOVORA IZ TEKSTA

Evaluacija modela sinteze govora vrši srednjom vrijednošću mišljenja slušatelja (MOS). MOS je skala od 1 od 5 sa korakom 0.5 te pripadajućim značenjima za svaku ocjenu [26]. Detaljni opis nalazi se u tablici 4.1.

**Tablica 4.1.** MOS skala [26]

| Ocjena | Značenje |
|---|---|
| 1 | Vrlo loša (bad) |
| 2 | Loša (poor) |
| 3 | U redu (fair) |
| 4 | Dobra (good) |
| 5 | Izvrsna (excellent) |

Formula za standardnu devijaciju označenu grčkim slovom σ, prikazana je jednadžbom 4.1. Veliko slovo N označava broj uzoraka u skupu, grčko slovo μ označava srednju vrijednost u skupu, a $x_i$ označava i-ti element skupa. Standardna pogreška aritmetičke sredine opisana jednadžbom 4.2. računa se kao standardna devijacija podijeljena sa kvadratnim korijenom broja uzoraka.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{4-1}$$

$$SPAS = \frac{\sigma}{\sqrt{N}} \tag{4-2}$$

## 4.1 Subjektivna ocjena kvalitete na testnom skupu

U anketi je sudjelovalo 10 slušatelja, odnosno ocjenjivača. Za slušatelje, odnosno ocjenjivače, napravljena je anketa koja sadrži 30 rečenica iz testnog skupa koji nije viđen od strane modela tijekom treninga, od kojih su 10 generirane modelom iz eksperimenta 2, 10 generiranih iz eksperimenta 4, a 10 su rečenice stvarnog govornika. Rečenice su posložene nasumično. Primjer takve rečenice je:

"Saborski zastupnik živog zida Ivan Pernar poznat je po problemima s logikom"

Rezultati ankete prikazani su u tablici 4.2. zajedno sa standardnom pogreškom aritmetičke sredine za rečenice.

**Tablica 4.2.** MOS rezultati prve ankete

| | **MOS** |
|---|---|
| Pročitane rečenice (engl. *ground truth*) | $4.64 \pm 0.12$ |
| Eksperiment 2 | $3.65 \pm 0.14$ |
| Eksperiment 4 | $3.95 \pm 0.16$ |

Iz rezultata je vidljivo da stvarne pročitane rečenice očekivano imaju najbolji rezultat. Eksperiment 4 polučio je bolji rezultat od eksperimenta 2. što je neočekivano s obzirom na prozodijski diverzitet podatkovnog skupa korištenog u eksperimentu 4. S druge strane, očekivano je da model eksperimenta 4 bolje generalizira, odnosno bolje izgovara riječi i rečenice koje nisu viđene tokom treninga.

## 4.2 Subjektivna ocjena kvalitete na skupu proizvoljnih rečenica

Skup podataka proizvoljnih rečenica sastoji se od 20 rečenica vijesti. Primjerice:

*"Dva američka bombardera B-52 danas su preletjela iznad Dubrovnika."*

Anketa se sastoji od 10 rečenica generiranih modelom eksperimenta 2 te 10 rečenica generiranih modelom eksperimenta 4. Rečenice su posložene nasumično. Rezultati ankete prikazani su u tablici 4.3. zajedno sa standardnom pogreškom aritmetičke sredine za rečenice.

**Tablica 4.3.** MOS rezultati druge ankete

| | **MOS** |
|---|---|
| Eksperiment 2 | $3.25 \pm 0.12$ |
| Eksperiment 4 | $3.75 \pm 0.14$ |

Model eksperimenta 4 polučio je bolji rezultat MOS skalom, dok mu je funkcija gubitaka na skupovima treninga i validacije lošija. Metrike modela ne utječu isključivo na sam ishod, već je i subjektivna ocjena vrlo važna pri ocjenjivanju generativnih modela općenito, pa tako i u ocjenjivanju TTS modela. Također, model eksperimenta 4 unatoč različitim prozodijskim izvorima, konverzacijskim podacima iz Podkast inkubatora te podacima iz informativnih emisija, neočekivano ostvaruje bolji MOS od modela iz jedinstvenog prozodijskog izvora. Očekivano, model eksperimenta 4 zbog veće količine i svog diverziteta u podatkovnom trening setu bolje generalizira rečenice od modela eksperimenta 2.

## 4.3 Analiza pogrešaka u sintetiziranim rečenicama

Od ispitanika se tražilo da sintetizirane snimke modela eksperimenta 2 i 4 analiziraju te prijave pogreške ili ukažu na nedostatke snimki. Izdvojene pogreške uključuju pogreške u naglašavanju te zvučna izobličenja, odnosno distorzije i šumove na lošije generiranim snimkama. Pogreške u naglašavanju prisutne su jer model nije naučio naglašavati pojedine riječi u određenim padežima, brojevima i rodovima što je specifičan problem za hrvatski jezik. Primjerice, model ne razlikuje homonime pa tako riječ gore (prijedlog) može biti naglašena kao riječ gore (množina od gora). Zvučna izobličenja, distorzije i šumovi produkt su nečistoća u podacima trening skupa, nedovoljnog treniranja vokoder modela, ili limita tehnologije kojom su realizirani modeli.

## 4.4 Analiza performansi

Performanse su mjerene na 10 rečenica dužina od 5 do 11 riječi te je uzet njihov prosjek. Primjer takve rečenice glasi:
*"Što je danas lijep i sunčan dan."*

Performanse modela sinteze govora dane su u tablici 4.4. Mjerenja brzine izvođenja izražene su u milisekundama. U tablici su dana mjerenja Tacotron 2 generatora spektrograma i HiFiGAN vokodera izvođenih na procesoru (engl. *Central Processing Unit - CPU*), odnosno grafičkoj kartici (engl. *Graphics Processing Unit - GPU*).

**Tablica 4.4.** Izmjerene performanse modela sinteze govora

| **Model** | **GPU** | **CPU** |
|---|---|---|
| Tacotron 2 | 354.32 | 14323.2 |
| HiFiGAN | 26.62 | 94.56 |

Tacotron 2 i HiFiGAN modeli, na grafičkoj kartici, pogodni su za primjenu u stvarnom vremenu. S druge strane, na procesoru, Tacotron 2 model nije ni približno pogodan za primjenu u stvarnom vremenu dok je HiFiGAN model i dalje podoban za primjenu u stvarnom vremenu. Brzina Tacotron 2 modela dodatno se može poboljšati smanjenjem brzine uzorkovanja, uz osjetni gubitak kvalitete zvučnog zapisa.

# 5. ZAKLJUČAK

U diplomskom radu obrađena je problematika sinteze teksta u govor putem modela dubokog učenja. Opisan je postupak kreiranja skupa podataka, opisane su tehnike prikupljanja, označavanja i obrade podataka. Opisano je rješenje modela kombinirajući Tacotron 2 arhitekturu za generator spektrograma te HiFiGAN arhitekture za generiranje zvučnog zapisa iz spektrograma. Detaljno su opisani provedeni eksperimenti treniranja modela. Modeli su odabrani za daljnju evaluaciju s obzirom na metrike funkcija gubitaka i grafova poravnanja, te su evaluirani od strane slušatelja pomoću anketa. Rezultati anketa prikazani su MOS skalom te je model implementiran u web aplikaciju realiziranu Streamlit programskim okvirom.

Aplikacija je uspješno realizirana. Sintetizator govora je tehnologija u razvoju. Trenutne mogućnosti tehnologije omogućuju sintezu ljudskog govora kvaliteti približnoj stvarnom govoru. Sinteza govora za hrvatski jezik moguća je, no još uvijek postoje problemi zbog specifičnosti samog jezika. Naglašavanja ne mogu biti lako riješena rječnikom zbog postojanja padeža, roda i broja. Također, homonimi su još jedan razlog zbog kojeg korištenje rječnika nije moguće, riječ vuci (kada nešto želimo vući) te vuci (arhaična množina riječi vuk) imaju drukčiji naglasak, a isti grafemski zapis. Takav problem može se riješiti robusnim jezičnim modelom kakav još uvijek ne postoji javno objavljen za hrvatski jezik, dok takvi slični modeli postoje za jezike većih govornih područja. Dodatno, model bi se mogao unaprijediti ručnim označavanjem naglasaka što iziskuje naprednije lingivističko znanje i poskupljuje dobivanje takvog podatkovnog skupa.

# LITERATURA

[1] History and Development of Speech Synthesis. (n.d.). SPA. Retrieved September 9, 2022, from http://research.spa.aalto.fi/publications/theses/lemmetty_mst/chap2.html [9.9.2022.]

[2] Oord AV, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K. Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499. 2016 Sep 12.

[3] Shen J, Pang R, Weiss RJ, Schuster M, Jaitly N, Yang Z, Chen Z, Zhang Y, Wang Y, Skerrv-Ryan R, Saurous RA. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In2018 IEEE international conference on acoustics, speech and signal processing (ICASSP) 2018 Apr 15 (pp. 4779-4783). IEEE.

[4] Łańcucki A. Fastpitch: Parallel text-to-speech with pitch prediction. InICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2021 Jun 6 (pp. 6588-6592). IEE

[5] Kong J, Kim J, Bae J. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. Advances in Neural Information Processing Systems. 2020;33:17022-33.

[6] Kim J, Kong J, Son J. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. InInternational Conference on Machine Learning 2021 Jul 1 (pp. 5530-5540). PMLR.

[7] W3C, W3C org, dostupno na: https://www.w3.org/ [9.9.2022.]

[8] Streamlit, Streamlit. Software Development. San Francisco, dostupno na: https://streamlit.io/ [9.9.2022.]

[9] YouTube, Alphabet, Inc, dostupno na: https://www.youtube.com/ [9.9.2022.]

[10] Audacity, The Audacity Team, dostupno na: https://www.audacityteam.org/ [9.9.2022.]

[11] Google Speech-to-Text, dostupno na: https://cloud.google.com/speech-to-text [9.9.2022.]

[12] Prodigy, dostupno na: https://prodi.gy/ [9.9.2022.]

[13] SoX, dostupno na: http://sox.sourceforge.net/ [9.9.2022.]

[14] PyTorch, dostupno na: https://pytorch.org/ [9.9.2022.]

[15] NVIDIA Tacotron 2 fork, dostupno na: https://github.com/BlazJurisic/tacotron2 [9.9.2022.]

[16] GitHub repositorij, dostupno na: https://github.com[9.9.2022.]

[17] HiFiGAN fork, dostupno na: https://github.com/BlazJurisic/hifi-gan [9.9.2022.]

[18] RTL direkt, RTL televizija, 2015, YouTube, https://www.youtube.com/playlist?list=PLCazT8AcMSrLbv6xfD4LksTifcR8N3leb [9.9.2022.]

[19] Stanje nacije, RTL televizija, 2022, YouTube, https://www.youtube.com/playlist?list=PLCazT8AcMSrKcVD180od647bG2dUcPQE4 [9.9.2022.]

[20] podcast inkubator intervju, https://www.youtube.com/watch?v=TB0o0VZ8eyU [9.9.2022.]

[21] LJSpeech dataset, dostupno na: https://keithito.com/LJ-Speech-Dataset/ [9.9.2022.]

[22] Battenberg E, Skerry-Ryan RJ, Mariooryad S, Stanton D, Kao D, Shannon M, Bagby T. Location-relative attention mechanisms for robust long-form speech synthesis. InICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2020 May 4 (pp. 6194-6198). IEEE.

[23] net.hr RSS feed, dostupno na: https://net.hr/rss [9.9.2022.]

[23] index.hr RSS feed, dostupno na: https://www.index.hr/rss/info [9.9.2022.]

[23] 24 sata RSS feed, dostupno na: https://www.24sata.hr/flatpages/rss [9.9.2022.]

[23] P. Kamp, Mean Opinion Score (MOS), Twilio Docs, 2020, https://www.twilio.com/docs/glossary/what-is-mean-opinion-score-mos [9.9.2022.]

# SAŽETAK

Ovaj diplomski rad bavi se problematikama izrade podatkovnog skupa za sintezu govora iz teksta, sintezom govora iz teksta te primjenom sinteze govora iz teksta u aplikaciji. Cilj rada je bio izraditi model dubokog učenja za sintezu govora na hrvatskom jeziku koristeći javno dostupne izvore govornika te realizirati aplikaciju za čitanje vijesti. Za potrebe učenja, prvo je izrađen podatkovni set postupcima prikupljanja podataka, uređivanja prikupljenih podataka, te označavanjem uređenih podataka. Model je realiziran metodom prijenosa učenja s engleskog na hrvatski jezik kroz četiri različita eksperimenta u kojima su mijenjani hiperparametri te podatkovni skupovi. Nakon realizacije modela, model je evaluiran korištenjem anketa te dodjeljivanjem subjektivnih ocjena, a analizirane su i greške koje je model proizvodio prilikom sinteze. Konačno, realizirana je web aplikacija koja koristi model na način da sintetizira audio snimke vijesti različitih portala iz teksta istih dobivenih RSS tokom.

**Ključne riječi:** duboko učenje, sinteza govora, Tacotron 2, HiFiGAN, web aplikacija za slušanje vijesti

# ABSTRACT

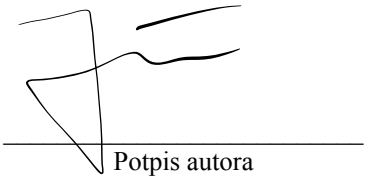
**News app realized with deep learning model**

This master thesis deals with the issues of building dataset for the speech synthesis from text, also, it deals with the issues of building speech synthesis models for croatian language and applying that model to an application for reading news. The goal of this thesis was to build a speaker dataset using public sources using data mining techniques, to process that data and to annotate it with linguistic features; transcripts of the speech the speaker has spoken. Model is realized with transfer learning from English to Croatian through 4 different experiments with different sets of hyperparameters and different datasets. After the model was built, it was evaluated through questionnaires in which subjects were giving their subjective opinion on the quality and realness of the generated speech. Also, they were asked to address mistakes they found during the listening of generated speech. Finally, a web app was built for the purpose of reading the news from Croatian web news portals using public RSS feeds from which text of the news was gathered.

**Keywords:** deep learning, speech and voice synthesis, Tacotron 2, HiFiGAN, PyTorch, web app for hearing news

ŽIVOTOPIS

Autor diplomskog rada, Blaž Jurišić, student je smjera programskog inženjerstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. U sedam godina iskustva rada u tvrtkama, R&D timovima te vodeći tvrtku čiji je osnivač, stekao je iskustvo u grani umjetne inteligencije koja se bavi problematikom zvuka, govora i teksta. Trenutno se bavi kloniranjem govora u stvarnom vremenu, detekciji životinjskih vrsta iz analize spektra zvuka te detekcijom robota i ljudi u pozivnim centrima.



Potpis autora