# Automatsko prepoznavanje govora primenom CTC modela i beam-search dekodiranja

## 1. Uvod i pregled literature

Pre pojave neuronskih mreža, sistemi za prepoznavanje govora uglavnom su se oslanjali na Hidden Markov Models (HMM) i Gaussian Mixture Models (GMM), koji su imali ograničenu sposobnost da modeluju složene vremenske zavisnosti u govoru.
Sa razvojem dubokih neuronskih mreža (DNN), konvolutivnih (CNN) i rekurentnih arhitektura (RNN, LSTM, GRU), omogućeno je da modeli direktno uče odnose između akustičkih i jezičkih obrazaca iz podataka, bez ručne ekstrakcije karakteristika.

Automatsko prepoznavanje govora (Automatic Speech Recognition – ASR) predstavlja proces pretvaranja audio signala u tekst. Savremeni sistemi za prepoznavanje govora često se oslanjaju na neuronske mreže i tehnike kao što su Connectionist Temporal Classification (CTC), koja omogućava modelima da uče sekvencijalne podatke bez eksplicitne poravnate oznake između zvuka i teksta.

Na početku rada razmatrani su gotovi modeli sa platforme Hugging Face i transformeri, međutim cilj projekta bio je da se razume princip rada i implementira sopstveno rešenje koristeći TensorFlow ili PyTorch. Tokom razvoja projekta pokazalo se da javno dostupni audio skupovi često sadrže kratke, neujednačene i loše normalizovane uzorke, pa je odlučeno da se napravi sopstvena obrada i generisanje Mel-spektrograma.

CTC gubitak (Graves et al., 2006) omogućava modelu da sam nauči optimalno poravnanje između ulaznih spektrograma i sekvenci karaktera, dok beam-search dekodiranje omogućava izbor najverovatnije sekvence izlaza. Ovaj pristup je kasnije popularizovan u radovima kao što su Deep Speech (Hannun et al., 2014) i implementacijama u TensorFlow-u i PyTorch-u.

## 2. Opis rešenja

Rešenje je implementirano u Python 3 programskom jeziku korišćenjem TensorFlow 2.18.0 biblioteke i Google Colab okruženja.
Model je zasnovan na kombinaciji konvolutivnih slojeva (CNN) i dvosmernih rekurentnih slojeva (BiGRU), koji su se pokazali efikasnim u prepoznavanju vremenskih obrazaca u zvučnim signalima.
Ulazni audio podaci se pretvaraju u Mel-spektrogram dimenzija 128 × 1200, koji omogućava reprezentaciju energije signala po frekvencijskim opsezima prilagođenim

ljudskom uhu.
Ova ideja i implementacija zasnivaju se na uputstvima iz članka *"Learn to Build Your First Speech-to-Text Model Using Python"* (Analytics Vidhya, 2019), gde je detaljno objašnjen proces ekstrakcije Mel-spektrograma pomoću Librosa biblioteke.
Na osnovu tog pristupa, kreirana je sopstvena klasa MelSpectrogramDataset, koja automatski generiše spektrograme i enkodira tekstualne oznake u numeričke vrednosti.

Tokom razvoja, konsultovani su i različiti izvori iz zajednice – uključujući forume i diskusije na Stack Overflow i TensorFlow GitHub repozitorijumu – koji su pomogli pri rešavanju problema sa formatima tenzora, oblikovanjem dimenzija i implementacijom CTC gubitka.
Iako ovi izvori nisu formalne naučne publikacije, bili su ključni za razumevanje praktičnih aspekata obrade podataka i treniranja sekvencijalnih modela u TensorFlow okruženju.

Sam model koristi Batch Normalization radi stabilizacije učenja, dok se Dropout koristi za smanjenje prenaučenosti.
Na kraju, izlazni sloj koristi softmax funkciju sa dodatnim *blank* indeksom koji omogućava efikasno izračunavanje CTC gubitka.
Za treniranje je korišćen Adam optimizator sa eksponencijalnim opadanjem stope učenja (ExponentialDecay).
Bez ovakvog rasporeda stope učenja, model se često zaglavljivao na *blank* predikcijama, dok je uvođenjem ExponentialDecay funkcije počeo da uči pravilnije obrasce i fonetske sekvence.
Eksperimentisanjem sa brojem epoha, veličinom batch-a i parametrom *clipnorm*, postignuta je stabilnija konvergencija i postepeno smanjenje gubitka.

## 3. Eksperimentalni rezultati

Trening je započet sa prosečnim gubitkom oko 525, koji je postepeno opadao do vrednosti od približno 183 nakon 15 epoha.
U ranim fazama model je predviđao samo znakove *„th"*, dok su kasnije epohe donele šire varijacije poput *„sx"*, *„sz"*, *„z"* i sličnih slučajnih sekvenci.
Na kraju obuke model je stabilno generisao jednostavne rečenice poput *„the e e e"*, što pokazuje da je naučio osnovne obrasce engleskog jezika i uspešno usvojio principe fonetskog poravnanja, iako još uvek ne generalizuje dovoljno da bi davao gramatički ispravne rečenice.

Model je treniran u Google Colab okruženju, sa pristupom NVIDIA T4 GPU-u (16 GB VRAM).
Lokalno testiranje je pokušano na računaru sa **NVIDIA RTX 3060** grafičkom karticom (12 GB VRAM), ali zbog ukidanja direktne podrške za CUDA drajvere u novijim verzijama Windows-a i ograničenja u **WSL2** okruženju, TensorFlow GPU akceleracija nije mogla biti pokrenuta lokalno.

Zbog toga je trening i evaluacija obavljena u Colab-u, koji obezbeđuje optimalnu podršku za CUDA i cuDNN verzije kompatibilne sa TensorFlow 2.18.0.

Za evaluaciju modela korišćen je test skup koji sadrži 300 nasumično odabranih uzoraka iz korpusa.
Kvalitet modela procenjivan je indirektno putem vrednosti CTC gubitka i analize dekodovanih sekvenci.
Gubitak je u prvim epohama naglo opadao (525 → 350 → 260), a zatim se stabilizovao između 190–183, što ukazuje na delimičnu konvergenciju bez potpune generalizacije.

Radi poređenja, izvršeno je i testiranje istih spektrograma korišćenjem greedy dekodiranja, bez beam-search algoritma.
Greedy dekoder je u većini slučajeva davao kraće i manje konzistentne sekvence („th“, „t“, „s“), dok je beam-search pristup uspevao da pronađe stabilnije rezultate („the e e e“), iako i dalje sa greškama.
Ovo potvrđuje prednosti beam search metode u odnosu na jednostavnije strategije dekodiranja, što je u skladu sa nalazima iz literature (Hannun et al., 2014).

Efikasnost implementacije (merenjem vremena obrade po uzorku) pokazala je da inference traje prosečno oko 0.8–1.2 sekunde po uzorku dužine 2–3 sekunde audio zapisa, što je prihvatljivo za istraživački prototip.
Međutim, preciznost merenjem standardnih metrika (kao što su Word Error Rate – WER ili Character Error Rate – CER) nije računata, jer test korpus nije imao adekvatnu veličinu i anotacije za takvu evaluaciju.

Uzroci stagnacije oko gubitka ≈190 uključuju:
(1) ograničen broj uzoraka i dužinu rečenica,
(2) uniformnost korpusa,
(3) odsustvo dodatnog jezičkog modela (Language Model) koji bi kontekstualizovao sekvence tokom dekodiranja.
U budućem radu planirano je proširenje eksperimentalnog okruženja, testiranje na većim datasetima (kao što je LibriSpeech), i implementacija jednostavnog jezičkog modela (npr. trigram LM) radi poboljšanja tačnosti.

Pored tehničkih ograničenja, primećeno je i da rezultati mogu varirati zavisno od kvaliteta i trajanja zvučnih uzoraka. Kratki segmenti govora često dovode do prenaglašenog učenja početnih fonema, dok duži uzorci rezultuju stabilnijim dekodiranjem. Ovo ukazuje na potrebu za balansiranijim skupom podataka i kontrolom trajanja klipova prilikom treniranja.
Dodatno, učestalost pojedinih slova u jeziku može dovesti do pristrasnosti modela ka određenim karakterima, što je primećeno u dominaciji izlaza poput “th” i “e”. Ova pojava je u literaturi poznata kao *CTC alignment collapse* i tipično se rešava kombinovanjem CTC sa pažljivim jezičkim modelom ili pretrenom akustičkom mrežom.

## 4. Zaključak

Projekat je uspešno demonstrirao osnovni princip automatskog prepoznavanja govora zasnovanog na CTC gubitku i beam-search dekodiranju. Model je pokazao sposobnost učenja osnovnih fonetskih obrazaca, iako je performansa ograničena količinom i kvalitetom podataka. Ključni doprinos projekta je razumevanje procesa od obrade zvuka i generisanja spektrograma do implementacije CTC gubitka i dekodiranja.

Iako SpecAugment nije korišćen u ovoj implementaciji, ova metoda predstavlja potencijalno značajno unapređenje. SpecAugment funkcioniše tako što nasumično maskira određene delove vremenskog i frekvencijskog domena spektrograma, čime se povećava raznolikost trening podataka i otpornost modela na šum. Ova metoda bi se mogla lako ugraditi u postojeći proces obrade podataka unutar funkcije parse_example_train(), što bi trebalo da dovede do bolje generalizacije bez potrebe za većim datasetom.

U budućnosti, sistem bi mogao biti poboljšan uvođenjem jezičkog modela (npr. trigram LM), korišćenjem većih skupova podataka kao što je LibriSpeech, kao i integracijom mikrofonskog unosa radi testiranja u realnom vremenu.

## 5. Literatura


1. Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist Temporal Classification: Labeling Unsegmented Sequence Data with Recurrent Neural Networks. ICML.
2. Hannun, A. et al. (2014), Cornell univerzitet. Deep Speech: Scaling up end-to-end speech recognition. arXiv:1412.5567.
3. Park, D. S. et al. (2019). SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. Interspeech.
4. TensorFlow Audio Recognition Guide.
https://www.tensorflow.org/tutorials/audio/simple_audio
5. Learn to build first speech to text model in python
https://www.analyticsvidhya.com/blog/2019/07/learn-build-first-speech-to-text-model-python/
6. Stack Overflow diskusija (nažalost uklonjena): *"Understanding TensorFlow CTC batch cost dimensions."* https://stackoverflow.com/questions/52848600